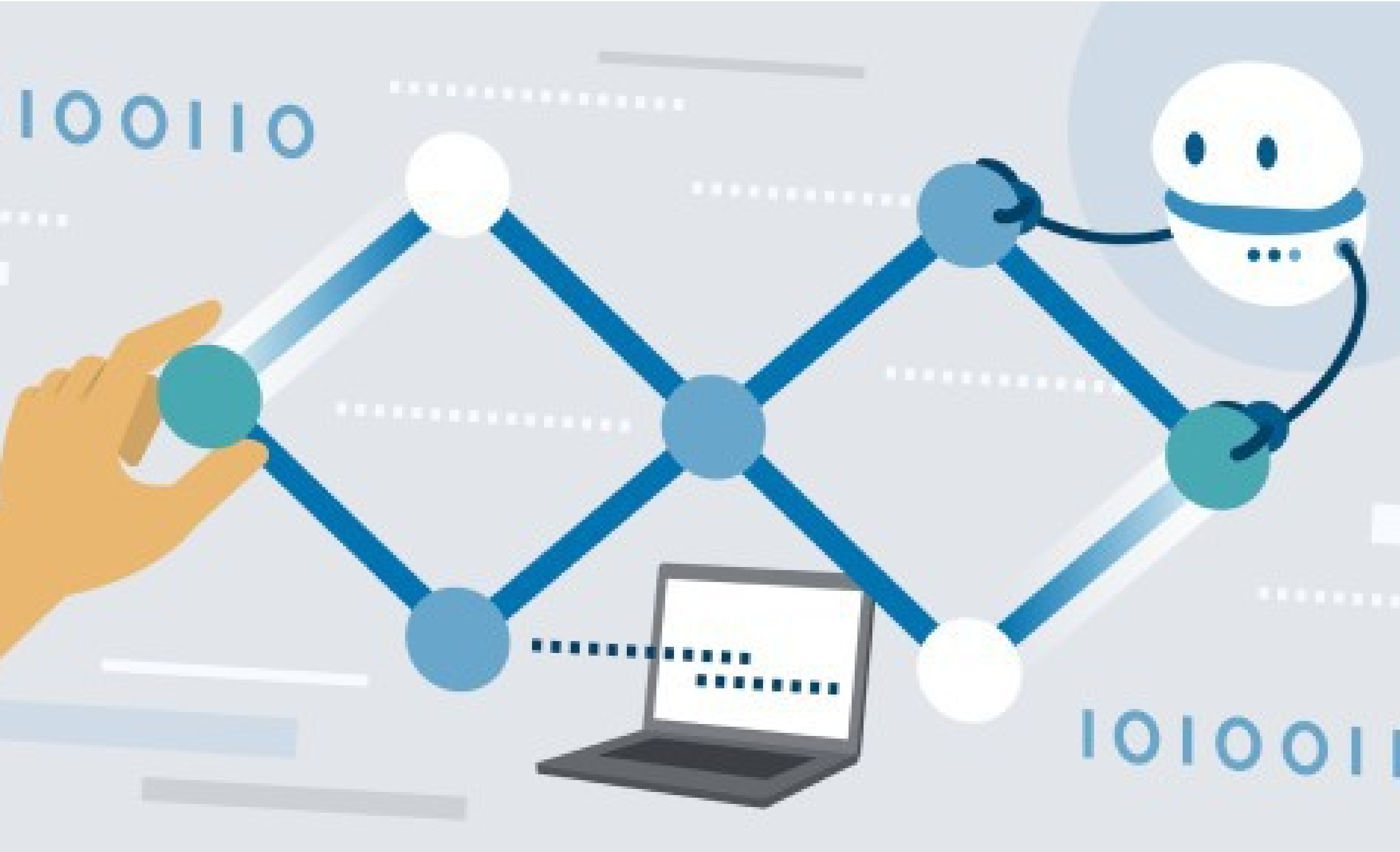
Data Science and MLOps Landscape in Industry



Introduction

unfold_lessHide cell

In [1]:

linkcode

```
# Import all the Python Libraries needed for the Exploratory Data Analysis
import pandas as pd
import numpy as np
import json
from collections import Counter

import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
import plotly.express as px
from plotly.offline import init_notebook_mode, iplot
from plotly.colors import n_colors

from IPython.core.display import display, HTML, Javascript
import IPython.display
from IPython.display import display, clear_output

import ipywidgets as widgets
```

```
from ipywidgets import interact, interact_manual


import matplotlib as mpl

import matplotlib.pyplot as plt


import warnings

warnings.filterwarnings('ignore')
```

/opt/conda/lib/python3.7/site-packages/geopandas/_compat.py:115: UserWarning: The Shapely GEOS version (3.9.1-CAPI-1.14.2) is incompatible with the GEOS version PyGEOS was compiled with (3.10.3-CAPI-1.16.1). Conversions between both will be slow.

shapely_geos_version, geos_capi_version_string

unfold_lessHide code

In [2]:

```
# Load the responses of the survey

df = pd.read_csv("../input/kaggle-survey-2022/kaggle_survey_2022_responses.csv")


# Get the questions' titles

questions_titles =  df[0:1]


# Skip the first row as it keeps the questions' titles

df = df[1:]
```

unfold_lessHide code

In [3]:

```
# Helper Functions for creating the visualizations in Plotly.


def create_scatter_plot(
    x_axis_values,
    y_axis_values,
    hover_template,
    marker_color,
    marker_size,
    title,
    subtitle,
    subtitle_explain):
    """It creates a Scatter Plot."""


    # Define the trace
    trace = go.Scatter(
        x=x_axis_values,
        y=y_axis_values,
        mode='markers',
        hovertemplate=hover_template,
        marker=dict(
            color=marker_color,
            size=marker_size,
```

```
        showscale=True,

        colorbar=dict(title="Percent"),

        opacity=0.7,

        colorscale = 'RdBu_r'

    )

)


# Define the layout

layout = go.Layout(

    width=900,

    height=950,

    plot_bgcolor="#fff",

    paper_bgcolor="#fff",

    showlegend = False,

    title = {

        'text' : f"<span style='font-size:30px; font-family:Times New Roman'>{title}</span><br><br><sup>{subtitle}</sup><br><sup>{subtitle_explain}</sup>",

        'x':0.5,

        'xanchor': 'center'

    },

    font = {"color" : '#7b6b59'},

    margin = dict(t=170),

)


fig = go.Figure(data = [trace], layout = layout)
```

```
fig.update_xaxes(
    showline=False,
    linewidth=1,
    linecolor='#c9c4c3',
    gridcolor='#c9c4c3',
    tickfont=dict(size=14, family='Verdana', color='#7b6b59'),
    title="",
    title_font=dict(size=14, family='Verdana', color='#f57369'),
    showgrid=False,
    tickangle=325
)
fig.update_yaxes(
    showline=False,
    linewidth=1,
    linecolor='#000',
    gridcolor='#fff',
    tickfont=dict(size=14, family='Verdana', color='#a43725'),
    title="",
    title_font=dict(size=14, family='Verdana', color='#f57369'),
    showgrid=False
)

fig.show()
```

```
def get_bar_plot_trace(x_values, y_values, display_text, top_n, rest_n, hovertext, orientation="h"):
    """It creates the trace for a bar plot."""

    trace = go.Bar(
        y = y_values,
        x = x_values,
        name = "",
        orientation = orientation,
        marker = dict(color = ["#E6b6a4"]*rest_n +  ["#a43725"]*top_n),
        text = display_text,
        texttemplate =  "<b style='color: #fff'>%{text}% </b>",
        textposition = ["outside"]*rest_n + ["inside"]*top_n,
        hovertext=hovertext
    )

    return trace


def create_single_bar_plot(x_values, y_values, display_text, top_n, rest_n, hovertext, title, subtitle="", orientation="h"):
    """It creates single bar plots."""

    trace = get_bar_plot_trace(x_values, y_values, display_text, top_n, rest_n, hovertext, orientation)
```

```
    large_title_format = f"<span style='font-size:30px; font-family:Times New Roman'>{title}</span>"

    layout = dict(
        title = large_title_format,
        font = dict(color = '#7b6b59'),
        margin = dict(t=120),
        yaxis={'categoryorder':'array','categoryarray': x_values},
        xaxis=dict(side="top", zerolinecolor = "#4d4d4d", zerolinewidth = 0.5, gridcolor="#e7e7e7",
tickformat=",.1%"),
        width = 800,
        height= 700,
        plot_bgcolor = "white"
    )

    fig = go.Figure(data = trace, layout = layout)
    fig.show()


def create_box_plot(df, x_column_name, y_column_name, title):
    """It creates bar plots."""

    fig = px.box(
        df,
        x=x_column_name,
```

```
        y=y_column_name,

        title=f"<span style='font-size:30px; color:#7b6b59; font-family:Times New Roman'>{title}</span>")


    layout = go.Layout(

        xaxis= {"title": ""},

        yaxis= {"title": "Compensation in USD"},

        font = dict(color = 'black'),

        paper_bgcolor='rgba(0,0,0,0)',

        plot_bgcolor='rgba(0,0,0,0)',

        height=800,

        width=1050

    )


    fig.update_layout(layout)

    fig.update_yaxes(showline=True, linewidth=1, gridcolor='lightgrey')

    fig.update_traces(marker_color='#b39a74')


    fig.show()


def create_heatmap(z, x, y, annotation_text, color_scale, title, subtitle="", xlabel="", ylabel=""):

    """It creates a heatmap."""


    fig = ff.create_annotated_heatmap(z, x=x, y=y, annotation_text=annotation_text,
colorscale=color_scale)
```

```
large_title_format = f"<span style='font-size:30px; font-family:Times New Roman'>{title}</span>"

small_title_format = f"<span style='font-size:14px; font-family:Helvetica'>{subtitle}</b></span>"



layout = dict(

    title = large_title_format + "<br>" + small_title_format,

    font = dict(color = '#7b6b59'),

     xaxis= {"title": xlabel},

    yaxis= {"title": ylabel},


)


fig['layout'].update(layout)

fig["layout"]["xaxis"].update(side="bottom")

fig.show()
```

unfold_lessHide code

In [4]:

```
# This section has all the python functions and global variables needed for the analysis



# Categorizing the state of Machine Learning Adoption into more general categories

map_ml_adoption = {

    "No (we do not use ML methods)": "Not Started" ,

    "We are exploring ML methods (and may one day put a model into production)": "Exploration Stage",
```

```
    "We use ML methods for generating insights (but do not put working models into production)": "Generating Insights",

    "We recently started using ML methods (i.e., models in production for less than 2 years)": "Models in Production",

    "We have well established ML methods (i.e., models in production for more than 2 years)": "Models in Production",

    "I do not know": "Not Known",

    np.nan: "Not Known"

}



# Colors for different Machine Learning Adoption Stages

ml_adoption_color_discrete_map={

    "Models in Production":"#a43725",

    "Generating Insights": "#c07156",

    "Exploration Stage":"#E6b6a4",

    "Not Started": "#e0d5bd",

    "Not Known": "#beb29e"

}



# Rephrasing the ML Adoption (state) by adding numbers for sorting them alphabetically

map_ml_usage = {

    "No (we do not use ML methods)": "0. Not Started<br><sup>(No ML)</sup>" ,

    "We are exploring ML methods (and may one day put a model into production)": "1. Exploration<br><sup>Only Exploring ML</sup>",
```

```
    "We use ML methods for generating insights (but do not put working models into production)": "2. Beginner Stage<br><sup>Use ML only for Insights</sup>",

    "We recently started using ML methods (i.e., models in production for less than 2 years)": "3. Intermediate Stage<br><sup>Recently Started Using ML</sup>",

    "We have well established ML methods (i.e., models in production for more than 2 years)": "4. Advance Stage<br><sup>Well Established ML</sup>",

    "I do not know": "Not Known",

    np.nan: "Not Known"

}



# Rephrasing the Company Size by adding numbers for sorting them alphabetically

map_company_size = {

    "0-49 employees": "1. 0-49 employees" ,

    "50-249 employees": "2. 50-249 employees",

    "250-999 employees": "3. 250-999 employees",

    "1000-9,999 employees": "4. 1000-9,999 employees",

    "10,000 or more employees": "5. 10,000 or more employees",

    np.nan: np.nan

}



# Rephrasing the Coding experience by adding numbers for sorting them alphabetically

map_programming_experience = {

    "I have never written code": "1. 0 years",

    "< 1 years": "2. < 1 years",
```

```
    "1-3 years": "3. 1-3 years",

    "3-5 years": "4. 3-5 years",

    "5-10 years": "5. 5-10 years",

    "10-20 years": "6. 10-20 years",

    "20+ years": "7. 20+ years",

    np.nan: np.nan

}




# Rephrasing the Machine Learning experience by adding numbers for sorting them alphabetically

map_ml_experience = {

    "I do not use machine learning methods": "1. 0 years",

    "Under 1 year": "2. < 1 years",

    "1-2 years": "3. 1-2 years",

    "2-3 years": "4. 2-3 years",

    "3-4 years": "5. 3-4 years",

    "4-5 years": "6. 4-5 years",

    "5-10 years": "7. 5-10 years",

    "10-20 years": "8. 10-20 years",

    "20+ years": "9. 20+ years",

    np.nan: np.nan

}
```

```python
# Rephrasing the Data Science Teams Size by adding numbers for sorting them alphabetically

map_data_team_size = {

    "0": "1. 0",

    "1-2": "2. 1-2",

    "3-4": "3. 3-4",

    "5-9": "4. 5-9",

    "10-14": "5. 10-14",

    "15-19": "6. 15-19",

    "20+": "7. 20+",

    np.nan: np.nan

}



# Get a plotly Dataset with all the countries along with the continent in which they belong

countries_df = px.data.gapminder().query("year == 2007")

countries_df["country"] = countries_df["country"].str.strip()


map_country_continent = {

    "United States of America": "Americas",

    "United Kingdom of Great Britain and Northern Ireland": "Europe",

    "South Korea": "Asia",

    "Russia": "Europe",

    "Viet Nam": "Asia",

    "Hong Kong (S.A.R.)": "Asia",
```

```python
    "Ukraine": "Europe",

    "United Arab Emirates": "Asia",

    "Iran, Islamic Republic of...": "Asia",


}



def fix_map_country_continent(map_countries: dict, country:str, continent:str):

    """It maps a country to its continent"""

    if country in map_countries:

        return map_countries[country]


    return continent




def usage_of_a_product_or_service(question_title: str, row: pd.Series, columns_list: list) -> str:

    """It takes as input a question title with multiple choices answers and checks

    if the respondent has selected at least one of the answers or not.

    For instance, if we want to check if a respondent uses cloud computing platforms, question 31, then we should

    check if the participant has selected any cloud computing platform choice Q31_1, Q31_2, etc.

    """

    for col in columns_list:

        if col.startswith(question_title):
```

```python
        if not pd.isnull(row[col]) and row[col].strip().lower() != "none":
            return "Yes"
    # If all the columns (choices), Q31_1, Q31_2, etc have empty values then the user hasn't selected
    # any platform so we return NO as the answer
    return "No"


def categorize_education(education:str) -> str:
    """Assigns more general categories to education levels."""
    if education in [
        "No formal education past high school",
        "Some college/university study without earning a bachelor’s degree"
    ]:
        return "Lower than Bachelor"

    if education == "Bachelor’s degree":
        return "Bachelor"

    if education == "Master’s degree":
        return "Master"

    if education in ["Doctoral degree", "Professional doctorate"]:
        return "Higher than Master"
```

```python
    return "Other"



def extract_and_count_all_the_multiple_choice_answers(question, df):
    """If we have a question with multiple choices it returns a data
    frame with the number of occurrences of each choice in the responses.
    """

    # e.g List of choices for Question, e.g. Q19 (computer vision methods)
    choices_list = [choice for choice in df.columns if choice.startswith(question)]
    dfs_list = []
    for col in choices_list:
        dfs_list.append(df.groupby([col]).agg({"Q2" : "count"}).reset_index().rename(columns={col: question, "Q2": "counts"}))

    agg_df = pd.concat(dfs_list)
    agg_df["relative_percent"] = agg_df.apply(lambda x : (x["counts"] / df.shape[0]), axis = 1)
    agg_df = agg_df.sort_values(by=["relative_percent"], ascending=True)

    return agg_df



def assign_label(service:str):
    """It returns the company name to which the product belongs.
    It takes care only of the 3 big techs: Google, Microsoft, Amazon.
```

```python
    """

    if "google" in service.lower():
        return "Google"

    if "aws" in service.lower() or "amazon" in service.lower():
        return "Amazon"

    if "azure" in service.lower() or "microsoft" in service.lower():
        return "Microsoft"

    if "ibm" in service.lower():
        return "IBM"

    return "Other"


def extract_the_number_of_responses(question_title: str, row: pd.Series, columns_list: list) -> str:
    """It takes as input an answer from a multiple-choice question and counts the number
    of respondents that have chosen it.
    """
    num_responses = 0
    for col in columns_list:
        if col.startswith(question_title):
```

```
        if not pd.isnull(row[col]):
            num_responses = num_responses + 1

    return num_responses



def wrap_df_text(df):
    return display(HTML(df.style.background_gradient(axis=0,  cmap='YlOrBr', subset=["Average number of selected choices"]).to_html().replace("\\n","<br>")))
```

unfold_lessHide code

In [5]:

```
# respondents that currently are not students (answer **No** the **Q5** question)
# currently are employed (They didn't answer the **Q23** question that "Currently not employed")
# have answered in what industry they are currently employed (or their most recent employer if retired) - **Q24 question has an answer**

scope_df = df[
    (df["Q5"] == "No") &
    (df["Q24"].notnull()) &
    (df["Q23"] != "Currently not employed")
]


# Assign more general categories to the state of Machine Learning Adoption in industry
```

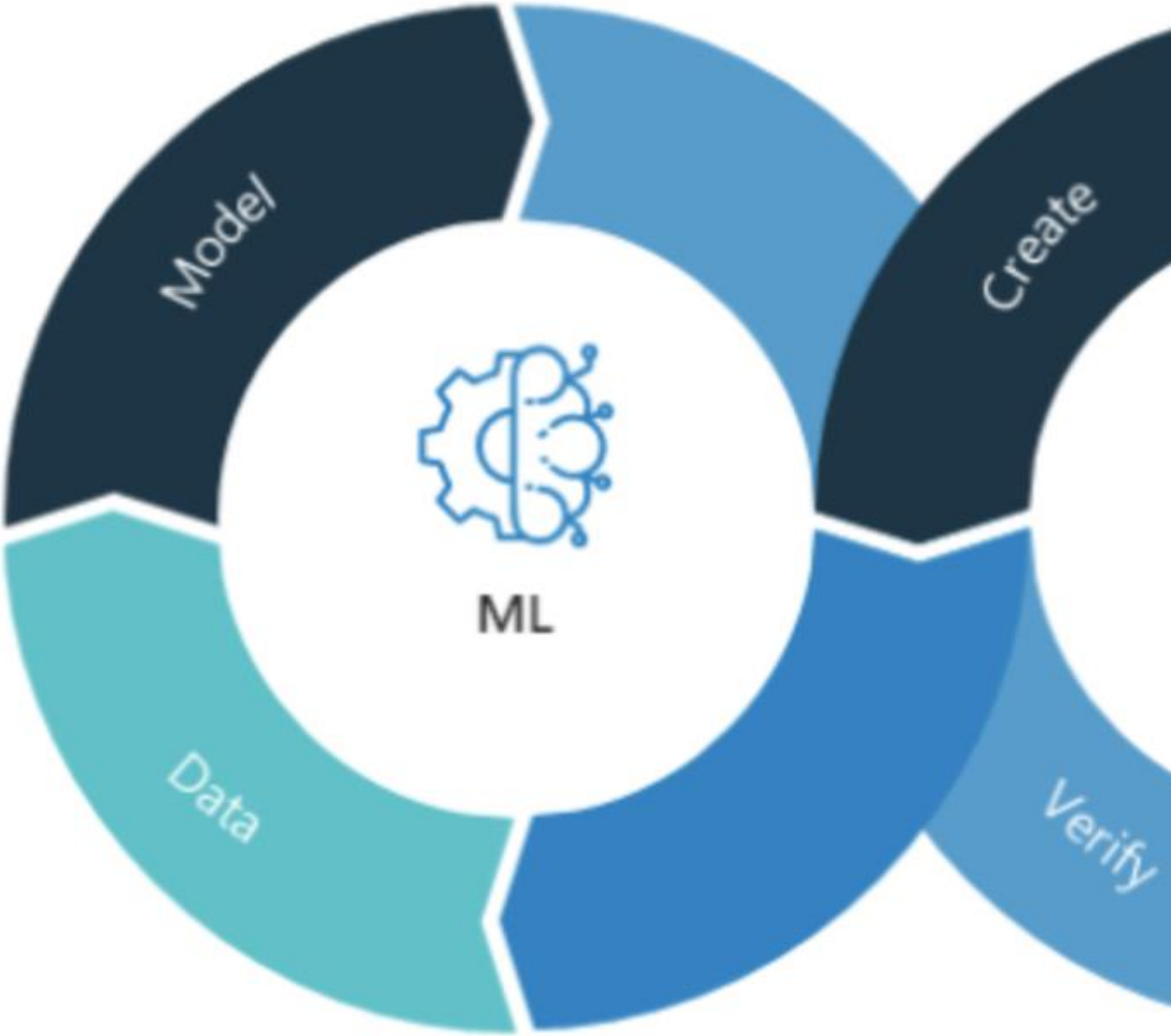
```
scope_df["ML_adoption_class"] = scope_df["Q27"].apply(lambda x : map_ml_adoption[x])


# Rephrasing the ML Adoption (state) by adding numbers for sorting them alphabetically

scope_df["ML_adoption"] = scope_df["Q27"].apply(lambda x : map_ml_usage[x])


# Rephrasing the size of the company by adding numbers for sorting them alphabetically

scope_df["Q25"] = scope_df["Q25"].apply(lambda x : map_company_size[x])


# Check if the respondent used Cloud Computing Platforms

scope_df["Cloud_usage"] = scope_df.apply(lambda row: usage_of_a_product_or_service("Q31", row, list(scope_df.columns)), axis=1)

scope_df["NLP_methods_usage"] = scope_df.apply(lambda row: usage_of_a_product_or_service("Q20", row, list(scope_df.columns)), axis=1)

scope_df["CV_methods_usage"] = scope_df.apply(lambda row: usage_of_a_product_or_service("Q19", row, list(scope_df.columns)), axis=1)

scope_df["GPU_usage"] = scope_df.apply(lambda row: usage_of_a_product_or_service("Q42", row, list(scope_df.columns)), axis=1)



scope_df["Q11"] = scope_df["Q11"].apply(lambda x : map_programming_experience[x])

scope_df["Q16"] = scope_df["Q16"].apply(lambda x : map_ml_experience[x])

scope_df["Q26"] = scope_df["Q26"].apply(lambda x : map_data_team_size[x])


industry_totals = scope_df["Q24"].value_counts().to_dict()
```

Adoption of Data Science and Machine Learning in Industry

As a Data Scientist in the banking sector, I strongly believe that the adoption of Data Science and Machine Learning could transform older, traditional banks into more digitally savvy banks capable of competing with the rise of more digitally-driven ones of the modern age. AI adoption can benefit other industries as well. The findings from the latest McKinsey Global Survey about the state of AI in 2021 indicate that AI adoption continues to grow and that the benefits remain significant. A majority of McKinsey survey respondents now say their organizations have adopted AI capabilities, as AI's impact on the bottom line is growing.

However, operationalizing and scaling machine learning to drive business value can be challenging. My experience has shown that, while many businesses have started diving into it, only a few data science projects actually make it to production. Moving from the experiment phase of ML to real-world deployment is difficult, as the journey requires finetuning ML models to fit the practical needs of a business and ensuring the solution can be implemented at scale.

ML Operationalization:



Source Nvidia Blog: What Is MLOps?

Models as part of an experiment are good, but models in production are great. MLOps, as the name implies, brings operationalization to the table, providing resources for bringing models from test environments into production.

Analysis's Target

The goal of this notebook is to extract insights from the responses of 2022 Kaggle Machine Learning & Data Science Survey about the state of AI Adoption and ML Operationalization in the industry in 2022 as well as about the Data Science landscape in the market. As I'm curious to see how the MLOps and AI adoption progressing in other organizations and what's the current trends in Data Science I'll try to enlighten the following main topics:

1. **What's the state of Machine Learning adoption in the enterprise today?**
    - What's the percentage of enterprises deploying data science and machine learning in production today?
    - Does the company's size or sector play a role in AI Adoption? Are larger companies more likely than smaller companies to have deployed AI in their organization?
2. **What's the enterprise AI tech stack?** The modern AI stack is a collection of tools, services, and processes imbibed with MLOps practices that allow developers and operations teams to build ML pipelines efficiently in terms of resource utilization, team efforts, end-user experience, and maintenance activities. It would be interesting if, for instance, we would answer the following questions:
    - Are Cloud-native solutions a must-have for business today?
    - What are the most popular tools for Data Storage, Data Management, AutoML, Business Intelligence, etc.?
    - What frameworks and libraries are commonly used in the market for Machine Learning and Data Science?
    - Are transfer learning methods mature enough to be used in the business environment?
    - Do we really work with big data and deep learning methods to such an extent that we need specialized hardware for ML models training?
3. **AI Careers & Job Outlook in 2022:**
    - What are the top AI job positions?
    - What does an AI professional do?
    - What are the professional AI skills in demand for 2022?
4. **AI Salary Overview**

Methodology

In order to have as much as I can a representative dataset for the analysis, I'll keep in the dataset only the professionals, namely the respondents that fulfill the criteria listed below:

- currently are not students (answer **No** the **Q5** question)
- currently are employed (They didn't answer "Currently not employed" to the **Q23** question)

- have answered in what industry they are currently employed (or their most recent employer if retired) **- Q24 question has an answer, not None**

As it can be seen below, ~ **37.9% of the total responses** meet the above criteria and the analysis will be conducted based on these responses.

unfold_lessHide code

In [6]:

```
mpl.rcParams.update(mpl.rcParamsDefault)

fig1 = plt.figure(figsize=(5,2),facecolor='white')

ax1 = fig1.add_subplot(1,1,1)
font = 'monospace'
ax1.text(0.9, 0.8, "Key figures",color='#7b6b59',fontsize=26, fontweight='bold', fontfamily=font, ha='center')

ax1.text(0, 0.4, "{:,d}".format(df.shape[0]), color='#e60000', fontsize=24, fontweight='bold', fontfamily=font, ha='center')
ax1.text(0, 0.001, "# of respondents \nin the survey",color='#757575',fontsize=15, fontweight='light', fontfamily=font,ha='center')

ax1.text(0.6, 0.4, "{}".format(scope_df.shape[0]), color='#e60000', fontsize=24, fontweight='bold', fontfamily=font, ha='center')
ax1.text(0.6, 0.001, "# of professionals",color='#757575',fontsize=15, fontweight='light', fontfamily=font,ha='center')
```

```
ax1.text(1.5, 0.4, "{}".format(round((scope_df.shape[0]/df.shape[0] )*100, 2))+"%", color='#e60000', fontsize=24, fontweight='bold', fontfamily=font, ha='center')

ax1.text(1.5, 0.001, "of the respondents are in the analysis \nscope",color='#757575',fontsize=15, fontweight='light', fontfamily=font, ha='center')


ax1.set_yticklabels('')

ax1.tick_params(axis='y',length=0)

ax1.tick_params(axis='x',length=0)

ax1.set_xticklabels('')


for direction in ['top','right','left','bottom']:

    ax1.spines[direction].set_visible(False)


fig1.subplots_adjust(top=0.9, bottom=0.2, left=0, hspace=1)


fig1.patch.set_linewidth(3)

fig1.patch.set_edgecolor('#E6b6a4')

fig1.patch.set_facecolor('white')

ax1.set_facecolor('white')


plt.show()
```

Key figures

23,997

# of respondents in the survey

9094

# of professionals

of the respo

Outlier Analysis

It would be also interesting to examine if there are some **"outlier respondents"** that have marked all the answers for the multiple-choice questions.

For that, I calculated the average number of choices that each respondent selected in the multiple-choice questions. I found out that each respondent selects 1 - 2 options in the multiple-choice questions on average.

Only 2% of the respondents in the scope have an average number of selections greater than 3, which cannot affect the results of the analysis. **Also, it doesn't necessarily mean that we have to address them as outliers. One explanation would be that they might have many years of coding or ML experience, so makes sense to be familiar with many frameworks and work with a variety of libraries.**

As the tables below illustrate, this hypothesis is valid since the biggest percentage of the respondents with more than 3 selections on average, have strong coding and machine learning experience.

So I won't discard these respondents or treat them differently.

unfold_lessHide code

In [7]:

```
# Collect all the multiple-choice questions
multiple_choice_questions = {}
seen_columns = []

for col in df.columns:
```

```
    question = col.split("_")[0]

    if question in seen_columns:

        if question not in multiple_choice_questions:

            multiple_choice_questions[question] = 2

        else:

            multiple_choice_questions[question] = multiple_choice_questions[question] + 1

    else:

        seen_columns.append(question)


# Create a new column in the dataframe for each of the multiple-choice questions which

# shows the number of the choices that the respondent selected for each one respectively.

for col in list(multiple_choice_questions.keys()):

    scope_df[f"{col}_number_of_responses"] = scope_df.apply(

        lambda x : extract_the_number_of_responses(col,x, df.columns), axis = 1)
```

unfold_lessHide code

In [8]:

```
respondents_mean_responses = scope_df[[f"{col}_number_of_responses" for col in list(multiple_choice_questions.keys())]].mean(axis = 1).reset_index().rename(columns={0: "Mean number of responses"})

#respondents_mean_responses["Mean number of responses"].mean()

# (respondents_mean_responses[

#     respondents_mean_responses["Mean number of responses"] > 3
```

```python
# ].shape[0]/scope_df.shape[0])*100


outliers = scope_df.filter(items=respondents_mean_responses[respondents_mean_responses["Mean number of responses"] > 3]["index"].to_list(), axis=0)

outliers = outliers.groupby(

    ["Q16"]

).agg(

    {"Q2" : "count"}

).reset_index().rename(

    columns={"Q2": "Nbr of respondents", "Q16": "Years of Machine Learning Experience"}

).sort_values(by=["Years of Machine Learning Experience"])

outliers["%"] = outliers.apply(lambda x : x["Nbr of respondents"] / outliers["Nbr of respondents"].sum(), axis = 1)

outliers["%"] = np.round(outliers["%"]* 100, 2)

outliers.style.background_gradient(axis=0,  cmap='YlOrBr', subset=["%"])
```

Out[8]:

	Years of Machine Learning Experience	Nbr of respondents	%
0	2. < 1 years	12	6.320000
1	3. 1-2 years	30	15.790000

2	4. 2-3 years	34	17.890000
3	5. 3-4 years	24	12.630000
4	6. 4-5 years	27	14.210000
5	7. 5-10 years	49	25.790000
6	8. 10-20 years	14	7.370000

unfold_lessHide code

In [9]:

```
outliers = scope_df.filter(items=respondents_mean_responses[respondents_mean_responses["Mean number of responses"] > 3]["index"].to_list(), axis=0)

outliers = outliers.groupby(

    ["Q11"]

).agg(

    {"Q2" : "count"}

).reset_index().rename(

    columns={"Q2": "Nbr of respondents", "Q11": "Years of Coding Experience"}

).sort_values(by=["Years of Coding Experience"])

outliers["%"] = outliers.apply(lambda x : x["Nbr of respondents"] / outliers["Nbr of respondents"].sum(), axis = 1)
```

```
outliers["%"] = np.round(outliers["%"]* 100, 2)

outliers.style.background_gradient(axis=0,  cmap='YlOrBr', subset=["%"])
```

Out[9]:

	Years of Coding Experience	Nbr of respondents	%
0	2. < 1 years	10	5.260000
1	3. 1-3 years	32	16.840000
2	4. 3-5 years	28	14.740000
3	5. 5-10 years	50	26.320000
4	6. 10-20 years	41	21.580000
5	7. 20+ years	29	15.260000

In the table below, we can also see the average number of choices that respondents selected for each of the multiple-choice questions and we might be able to conclude the following findings:

- The professionals who participated in the survey, use on average 2 programming languages on a regular basis, 3 Machine Learning Algorithms, and 2 Machine Learning Frameworks.

- In addition, they usually don't use natural language processing (NLP) methods like Word embeddings/vectors (GLoVe, fastText, word2vec), Encoder-decoder models (seq2seq, vanilla transformers), Contextualized embeddings, or Transformer language models

unfold_lessHide code

In [10]:

```
outlier_analysis = []


for col in list(multiple_choice_questions.keys()):

    mean_responses = round(scope_df[f"{col}_number_of_responses"].mean())

    outlier_analysis.append([

        col,

        multiple_choice_questions[col],

        mean_responses,

    ])


average_responses = pd.DataFrame(outlier_analysis, columns = ["Question", "Nbr of available Choices", "Average number of selected choices"])

average_responses["Question Title"] = questions_titles[[f"{col}_1" for col in list(multiple_choice_questions.keys())]].loc[0].to_list()

average_responses["Question Title"] = average_responses["Question Title"].apply(lambda x : x.split("(Select")[0].strip())


#Updates the DataFrame in place

scope_df.drop([f"{col}_number_of_responses" for col in list(multiple_choice_questions.keys())], axis = 1, inplace=True)
```

```
average_responses["Question Title"] = average_responses['Question Title'].str.wrap(80)

average_responses = average_responses[["Question", "Question Title", "Nbr of available Choices",
"Average number of selected choices"]]

wrap_df_text(average_responses)
```

	Question	Question Title	Nbr of available Choices	Av sel
0	Q6	On which platforms have you begun or completed data science courses?	12	2
1	Q7	What products or platforms did you find to be most helpful when you first started studying data science?	7	2
2	Q10	Did your research make use of machine learning? - Yes, the research made advances related to some novel machine learning method (theoretical research)	3	0
3	Q12	What programming languages do you use on a regular basis?	15	2
4	Q13	Which of the following integrated development environments (IDE's) do you use on a regular basis?	14	3
5	Q14	Do you use any of the following hosted notebook products?	16	1
6	Q15	Do you use any of the following data visualization libraries on a regular basis?	15	2

7	Q17	Which of the following machine learning frameworks do you use on a regular basis?	15	2
8	Q18	Which of the following ML algorithms do you use on a regular basis?	14	3
9	Q19	Which categories of computer vision methods do you use on a regular basis?	8	1
10	Q20	Which of the following natural language processing (NLP) methods do you use on a regular basis?	6	0
11	Q21	Do you download pre-trained model weights from any of the following services?	10	1
12	Q28	Select any activities that make up an important part of your role at work:	8	2
13	Q31	Which of the following cloud computing platforms do you use?	12	1
14	Q33	Do you use any of the following cloud computing products?	5	1
15	Q34	Do you use any of the following data storage products?	8	1
16	Q35	Do you use any of the following data products (relational databases, data warehouses, data lakes, or similar)?	16	1

17	Q36	Do you use any of the following business intelligence tools?	15	1
18	Q37	Do you use any of the following managed machine learning products on a regular basis?	13	1
19	Q38	Do you use any of the following automated machine learning tools?	8	1
20	Q39	Do you use any of the following products to serve your machine learning models?	12	1
21	Q40	Do you use any tools to help monitor your machine learning models and/or experiments?	15	1
22	Q41	Do you use any of the following responsible or ethical AI products in your machine learning practices?	9	1
23	Q42	Do you use any of the following types of specialized hardware when training machine learning models?	9	1
24	Q44	Who/what are your favorite media sources that report on data science topics?	12	3

Ready to move on to the next sections of the Deep Dive Analysis? □

Table of Contents

What's the state of Machine Learning adoption in the enterprise today?

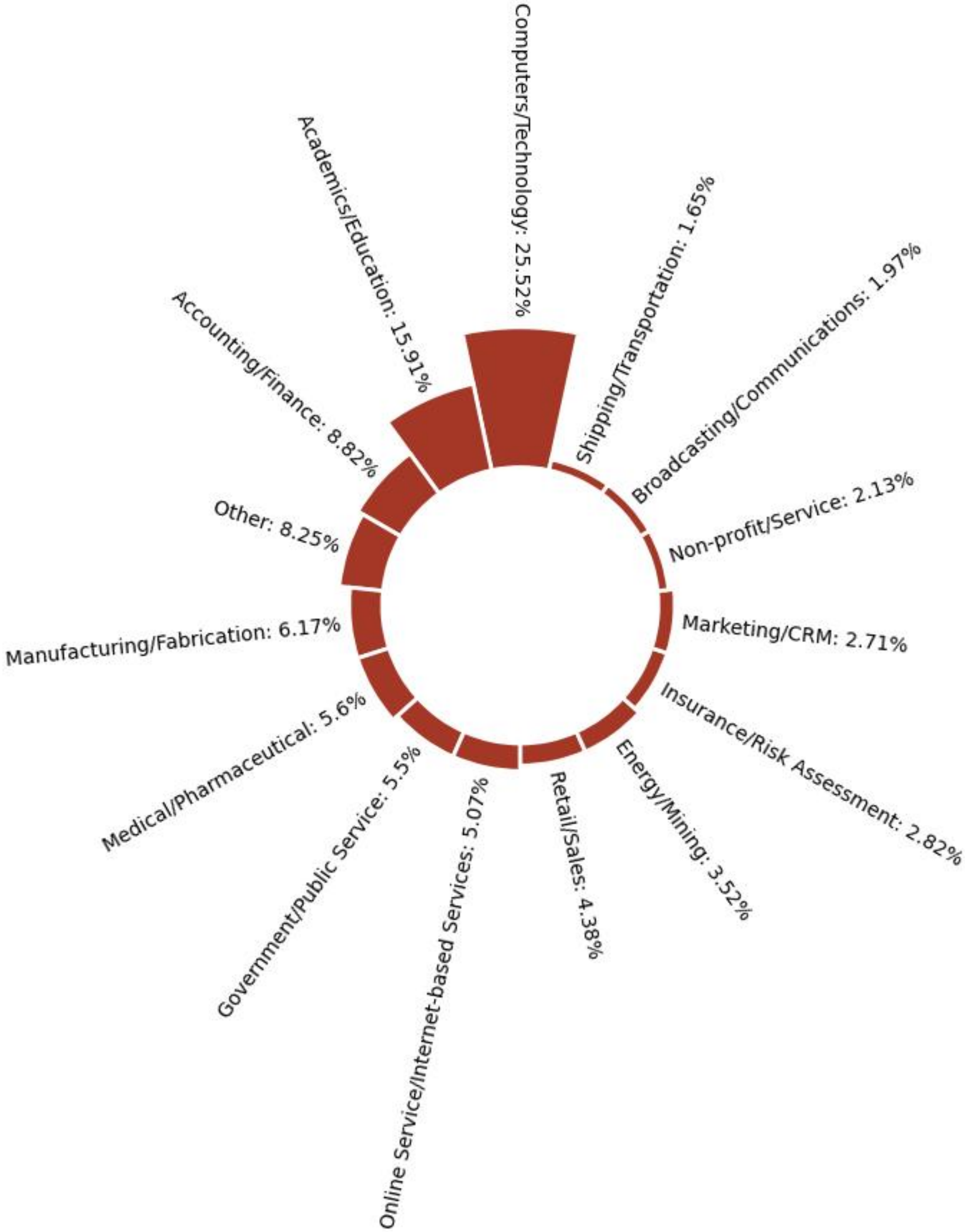
The first thing that I want to understand from the survey responses, is the state of ML adoption in different industries today. In the 2022 Kaggle Machine Learning & Data Science survey of 9,094 professionals coming from different industries, as it can be seen in the chart below,

- a percentage of 25.52% working in tech companies,
- a 15.91% in the academic field,
- and the rest distributed from the finance sector to shipping and transportation.

Which sector would you bet is a high performer in AI and has made big progress in terms of AI adoption?

Before I answer that, let's see how AI adoption looks like broadly, across all sectors, in 2022.

unfold_moreShow hidden code


Computers/Technology: 25.52%
Shipping/Transportation: 1.65%
Broadcasting/Communications: 1.97%
Non-profit/Service: 2.13%
Marketing/CRM: 2.71%
Insurance/Risk Assessment: 2.82%
Energy/Mining: 3.52%
Retail/Sales: 4.38%
Online Service/Internet-based Services: 5.07%
Government/Public Service: 5.5%
Medical/Pharmaceutical: 5.6%
Manufacturing/Fabrication: 6.17%
Other: 8.25%
Accounting/Finance: 8.82%
Academics/Education: 15.91%

The data shows that about ~ **33%** of respondents say that their organizations have Machine Learning models in production, either in an advanced stage or in an intermediate stage (they recently started using ML methods), while a percentage of **10.2%** uses ML methods for generating insights. However, a considerable percentage of the participants, **21.7%**, answered that their companies haven't started yet using AI and ML techniques while **17.1%** of the respondents say that have started exploring the capabilities of this new technology.

unfold_moreShow hidden code

32.8%21.7%18.3%17.1%10.2%

Models in ProductionNot StartedNot KnownExploration StageGenerating InsightsThe State of the ML Adoption in Inudstry in 2022

Models in ProductionNot StartedNot KnownExploration StageGenerating Insights4. Advance StageWell Established ML3. Intermediate StageRecently Started Using ML0. Not Started(No ML)Not Known1. ExplorationOnly Exploring ML2. Beginner StageUse ML only for Insights

Now, let's come back to the question above and try to answer it by extracting some insights from the survey results.

It is clear in the following chart that companies providing **Internet-based services** have a **better adoption of Machine Learning and Data Science** followed by **Insurance companies**, whereas **non-profit organizations and the government sector score undoubtedly lower for the adoption of various AI-related technologies.** A key reason for the lower AI adoption among governments and non-profit organizations is the bureaucracy and the established processes that take too long. In these sectors, might be less encouragement for employees to take risks and innovate.

In the private sector, employers tend to put a strong focus on experimentation, innovation, and growth. For instance, **companies providing Internet-based services** could gather many data from the user's online activities and the employees can apply analytics and other innovative ideas in order to improve the services that their company provides. The **insurance sector** is also leveraging AI technologies for insurance advice, underwriting claims processing, fraud prevention, risk management, and direct marketing. Customer behavior and advances in technology have opened the door for AI in the insurance market to create value, reduce costs, increase efficiency and achieve higher customer satisfaction and trust. **Retail** has also embraced AI technologies, with 27% of the professionals working in the retail sector, saying their companies have well-established machine learning methods in production.

unfold_moreShow hidden code

Academics/EducationAccounting/FinanceBroadcasting/CommunicationsComputers/TechnologyEnergy/MiningGovernment/Public ServiceInsurance/Risk AssessmentManufacturing/FabricationMarketing/CRMMedical/PharmaceuticalNon-profit/ServiceOnline Service/Internet-based ServicesOtherRetail/SalesShipping/Transportation Not Started(No ML) Exploration Only Exploring ML Beginner StageUse ML only for Insights Intermediate StageRecently Started Using ML Advance StageWell Established ML

101520253035PercentThe State of Machine Learning Adoption by IndustryQuestions Data: Industry (Q24) and ML Adoption State (Q27)Size,Color: Percentage of Respondents - The number of respondents of the related sector that chose the relevant adoption stage of their company divided by the total number of respondents working in that sector.

unfold_moreShow hidden code

1. 0-49 employees2. 50-249 employees3. 250-999 employees4. 1000-9,999 employees5. 10,000 or more employees0100200300400500600700800900

Models in ProductionExploration StageGenerating InsightsNot StartedProductionization of ML models by Company's size

Another important insight that comes up from the analysis is that big companies are leading the way in AI adoption.

The survey results show that larger companies, with 1000-9,999 employees or more than 10,000 are the leading AI adopters. There are several reasons that may explain why larger companies outpace smaller ones in AI adoption. For one, because large firms tend to serve large markets, they can better amortize the high fixed costs associated with employing AI production technologies over more sales. In addition to that, larger firms offer higher wages and more benefits, increasing the pool of top AI talent these firms have access to. Finally, because vendors of AI systems benefit from supplying companies with the largest consumer base, vendors may focus on creating relationships and contracts with larger firms, enabling these firms to be more exposed to the value AI systems can bring to their businesses.

In the next section, I'll explore tools and practices used in the market, according to the survey responses to establish an adaptable infrastructure for Machine learning and Data Science projects.

Overview of the enterprise AI technology stack

Machine learning was mainly in the experimental stage in the enterprise market not long ago. The Data Science teams always start with a Proof Of Concept (POC) approach and eventually gain traction even with a non-standardized production deployment process because of the business results achieved by the model. In order to scale this solution successfully with re-usability and reliability, the AI stack requires hardware and software optimizations in architectural areas of computing, memory, and networking.

Usage of Cloud Computing Platforms

According to several reports about the Cloud Computing Market in 2022, the adoption of cloud technologies continues to accelerate. Cloud computing has influenced the rise of machine learning and artificial intelligence. Factors such as affordable storage, availability of GPUs, faster AI training and inferencing performance, lower costs, and protection against attacks made machine learning accessible and affordable to businesses. Most companies lack the infrastructure and expertise to implement AI applications themselves.

As the following radar chart depicts, companies that have models in production use also cloud computing platforms which is reasonable since the cloud makes it easy for enterprises to experiment with machine learning capabilities and scale up as projects go into production and demand increases.

unfold_moreShow hidden code

Out[15]:

	Usage of Cloud Computing Platforms	Nbr of respondents	%

0	No	4994	54.920000
1	Yes	4100	45.080000

unfold_moreShow hidden code

0. Not Started(No ML)1. ExplorationOnly Exploring ML2. Beginner StageUse ML only for Insights3. Intermediate StageRecently Started Using ML4. Advance StageWell Established MLNot Known0200400600800100012001400

Cloud Usage: YesCloud Usage: NoCloud Usage by ML Adoption

Which cloud computing platforms are used for Machine Learning operations?

In the following visualizations, we can see the most popular cloud computing platforms by sector as well as by country. It is immediately obvious that Amazon Web Services (AWS) and Google Cloud Platform (GCP) are the dominant ones as well as that Alibaba Cloud is quite famous in Asia.

unfold_moreShow hidden code

Amazon Web Services (AWS) Microsoft Azure Google Cloud Platform (GCP) IBM Cloud Oracle Cloud SAP Cloud VMware Cloud Alibaba Cloud Tencent Cloud Huawei Cloud Academics/EducationAccounting/FinanceBroadcasting/CommunicationsComputers/TechnologyEnergy/MiningGovernment/Public ServiceInsurance/Risk AssessmentManufacturing/FabricationMarketing/CRMMedical/PharmaceuticalNon-profit/ServiceOnline Service/Internet-based ServicesOtherRetail/SalesShipping/Transportation

5101520253035PercentCloud Computing In Different IndustriesQuestions Data: Industry (Q24) and Cloud Computing Platform (31)Size,Color: Percentage of Respondents - The number of respondents of the related sector that chose the relevant Cloud Computing Platformdivided by the total number of respondents working in that sector.

unfold_moreShow hidden code

Google Cloud Platform (GCP)Amazon Web Services (AWS)Microsoft AzureAlibaba CloudMost Popular Cloud Computing Platform by Country

Machine Learning tools & products popular in 2022

The following graphs summarize the usage patterns of other tools, techniques, databases, platforms, and frameworks used by professionals.

unfold_moreShow hidden code

Note: The following chart is interactive, Click on the Clusters to view more details

Data Products: %Cloud Computing Platforms: %BI Tools: %Data Storage Products: %Cloud Computing Products: %ML Products: %Auto ML: %

Each company has a unique technology stack with software that they prefer to use with their proprietary data. There are a number of different platforms that go into each category of the stack. These categories include Visualization & Analytics, Computation, Storage Distribution & Data Warehouses. There are too many platforms to count, but in the following illustration, I'll be going over the popular cloud computing services and products that I have seen across the survey responses, offered by the top 4 giant Tech Companies: Amazon, Google, Microsoft & IBM.

- **Amazon top products:**
    - The most commonly used product provided by Amazon is **Amazon Web Services (AWS)** cloud computing platform, as it is used by 2346 respondents out of 9094 (**25.8%** of the professionals).
    - The second most popular is the Amazon Simple Storage Service (S3) as it's used by 17.8% of the respondents in the scope.
- **Google top products:**

    - As above, the most popular product offered by Google is its cloud computing platform, **Google Cloud Platform (GCP)**, used by **22.6%** of the respondents.
    - Secondly comes the Google Cloud Compute Engine which is slightly more popular than the Google Cloud Storage.
- **Microsoft top products:** The Microsoft products that dominate in the market according to the survey respondents' choices are Microsoft Power BI (18.23% of the responses in scope) and **Microsoft Azure** (used by **15.57%** of the respondents), and so it ranks 3rd in the list with the top cloud computing platforms (1st: AWS, 2nd: GCP).
- **IBM top products:** From IBM products, the IBM Watson Studio, followed by the IBM Cloud / Red Hat has gained the most popularity.

NOTES:

- The size of the rectangles in the third level of the treemap indicates the number of respondents using the relevant product/service, while the size of the rectangles and the counts respectively in the second level doesn't correspond to the number of respondents using Amazon, Google, etc. in general. The counts of each of the 4 companies in the second level of the map are just the sum of the respondents that use each of their services/products in the 3rd level. However, if the same user uses two or more products, provided by the same company it will be counted twice in the total sum of the second level. That's why the counts in the second level should not be taken into account as they do not represent the accurate total number of respondents that use them (it's a higher number than expected).
- The color of the rectangles in the third level of the treemap indicates the percentage of the respondents using the relevant product/service and it is applied the same logic as above.

unfold_moreShow hidden code

AI Tech StackAmazonGoogleMicrosoftIBM Amazon Web Services (AWS) Amazon Simple Storage Service (S3) Amazon Elastic Compute Cloud (EC2) Amazon SageMaker Amazon RDS Amazon Sagemaker Studio Amazon Elastic File System (EFS) Amazon Redshift Amazon DynamoDB Amazon Sagemaker Studio Lab Amazon Sagemaker Autopilot Amazon QuickSight Amazon AI Ethics Tools (Clarify, A2I, etc) Amazon EMR Notebooks Google Cloud Platform (GCP) Google Cloud Compute Engine Google Cloud Storage (GCS) Google Cloud BigQuery Google Data Studio Google Cloud Filestore Google Cloud AutoML Google Cloud SQL Google Cloud Vertex AI Workbench Google Cloud Vertex AI Google Responsible AI Toolkit (LIT, What-if, Fairness Indicator, etc) Microsoft Power BI Microsoft Azure Microsoft SQL Server Microsoft Azure Virtual Machines Microsoft Azure Blob StorageMicrosoft Azure SQL Database Microsoft Azure Files Azure Notebooks Azure Machine Learning Studio Azure Automated Machine Learning Microsoft Responsible AI Resources (Fairlearn, Counterfit, InterpretML, etc) Microsoft Azure Synapse IBM Watson Studio IBM Cloud / Red Hat IBM Db2 IBM AI Ethics tools (AI Fairness 360, Adversarial Robustness Toolbox, etc

0.050.10.150.20.25relative_percent

Frameworks, libraries and languages for Machine Learning & Data Science

unfold_moreShow hidden code

1.51% 1.75% 5.6% 7.25% 9.09% 11% 11.39% 13.28% 13.72% 15.02% 21.23% 47.35% 79.9% 00.20.40.60.8JuliaGoPHPC#MATLABBashCJavaC++JavascriptRSQLPython6.72% 7.63% 9.17% 9.59% 13.01% 17.65% 17.78% 19.94% 23.71% 24.6% 40.84% 60.88% 00.20.40.6IntelliJ MATLAB Vim / Emacs Sublime Text Spyder RStudio Visual Studio Notepad++ JupyterLab PyCharm Visual Studio Code Jupyter Notebook

Top programming languages for Data Science & ML in 2022Python Is Essential for Data Analysis and Data Science.The length of the bars denotes the percentage of professionals that use the relevant language.The counts are also visible by hover.

When it comes to the programming languages, the bar plot shows that Python is the most popular language followed by SQL and R.

- **Python** is the dominant language in the Machine Learning and Data Science field with 79.9% of the professionals using it for their daily tasks. Python is widely used in the industry, and it is also by far the language most recommended to beginners.
- **SQL** is necessary required when working with databases. Having at least a basic understanding of SQL and database management would go a long way in your career.
- **R:** a percentage of 21.2% of the respondents working in industry use R. While in most cases Python is the default choice when analyzing data and applying statistical methods, R is preferred as we'll see in a later section by many statisticians.

unfold_lessHide code

In [22]:

```
data_viz_libs = extract_and_count_all_the_multiple_choice_answers("Q15", scope_df)

data_viz_libs["relative_percent"] = round(data_viz_libs["relative_percent"] * 100,2)

data_viz_libs = data_viz_libs.rename(

    columns={"Q15":"Data Visualization Libraries", "counts": "# of respondents", "relative_percent": "% of respondents"}

)

data_viz_libs = data_viz_libs.sort_values(by=["% of respondents"], ascending=False).reset_index(drop=True)




ml_frameworks = extract_and_count_all_the_multiple_choice_answers("Q17", scope_df)

ml_frameworks["relative_percent"] = round(ml_frameworks["relative_percent"] * 100,2)

ml_frameworks = ml_frameworks.rename(

    columns={"Q17":"ML Frameworks", "counts": "# of respondents", "relative_percent": "% of respondents"}

)

ml_frameworks = ml_frameworks.sort_values(by=["% of respondents"], ascending=False).reset_index(drop=True)




colors = n_colors('rgb(230, 182, 164)', 'rgb(164, 55, 37)', 15, colortype='rgb')

a = [14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]




fig = make_subplots(
```

```
    rows=1, cols=2,
    #shared_xaxes=True,
    vertical_spacing=0.03,
    specs=[[{"type": "table"}, {"type": "table"}],
          ]

)

fig.add_trace(
go.Table(
  header=dict(
    values=["Data Visualization Libraries", "% of respondents"],
    line_color='white', fill_color='white',
    align='center', font=dict(color='black', size=12)
  ),
  cells=dict(
    values=[data_viz_libs["Data Visualization Libraries"], data_viz_libs["% of respondents"]],

                        fill_color=[np.array(colors)[a]],
    align='center', font=dict(color='white', size=13, family='Arial Rounded MT Bold')
  )),
    row=1, col=1
)
fig.add_trace(
```

```
go.Table(
  header=dict(
    values=["ML Frameworks", "% of respondents"],
    line_color='white', fill_color='white',
    align='center', font=dict(color='black', size=12)
  ),
  cells=dict(
    values=[ml_frameworks["ML Frameworks"], ml_frameworks["% of respondents"]],

                        fill_color=[np.array(colors)[a]],
    align='center', font=dict(color='white', size=13, family='Arial Rounded MT Bold')
  )),
    row=1, col=2
)

large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Top Data Visualization Libraries and ML Frameworks</span>"
small_title_format = "<span style='font-size:14px; font-family:Helvetica'></b></span>"



fig.update_layout(
    height=600,
    font = dict(color = '#7b6b59'),
    showlegend=False,
    title = large_title_format + "<br>" + small_title_format,
```

)

fig.show()

Matplotlib Seaborn Plotly / Plotly Express Ggplot / ggplot2 None Shiny Geoplotlib Bokeh D3 js Leaflet / Folium Other Altair Pygal Highcharter Dygraphs Data VisualizationLibraries64.1949.9727.5720.9513.466.394.964.664.333.363.111.681.1410.88% of respondents Scikit-learn TensorFlow Keras Xgboost PyTorch LightGBM Huggingface CatBoost None PyTorch Lightning Caret Fast.ai Other Tidymodels JAX ML Frameworks57.5237.4231.7826.7126.0813.038.717.136.325.285.173.63.313.281.07% of respondents

Top Data Visualization Libraries and ML Frameworks

An important task in Data Science is representing information in a visual context. **How can you make it easy to understand real-time trends and business insights present in the data?**

The answer is ... **Data Visualizations!!!**

Can you believe that the human brain takes only 13 milliseconds to process an image?

Humans love stories, and visualizations allow us to create one from data. Understanding data requires the use of data visualizations, and this is because visuals are processed 60,000 times faster than text inside the human brain. Using charts or graphs to visualize vast amounts of complex information is more straightforward than digging spreadsheets or reports.

The table above at the left provides the top Data Visualization Libraries that are excellent choices for creating visually appealing and insightful data representations according to the survey respondents, with the top-end respondents mainly preferring and using the originals **Matplotlib**, **Seaborn**, and **Plotly**, with **Ggplot** for R.

Without surprising us, the top Machine Learning Frameworks are **Scikit-learn**, followed by **Tensorflow** and **Keras** which are usually used for productionizing Deep Learning Models. Both frameworks are user-friendly and they provide high-level APIs for building and training models easily.

unfold_lessHide code

In [23]:

```
dfs_list = []

for col in [column for column in df.columns if column.startswith("Q18")]:
    dfs_list.append(scope_df.groupby([col]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts", col: "ML Algorithms"}))

ml_algorithms = pd.concat(dfs_list)
ml_algorithms["relative_percent"] = ml_algorithms.apply(lambda x : x["counts"] / scope_df.shape[0], axis = 1)

ml_algorithms = ml_algorithms.sort_values(by=["relative_percent"], ascending=True)
ml_algorithms = ml_algorithms[~ml_algorithms["ML Algorithms"].isin(["None", "Other"])]

create_single_bar_plot(
    x_values=ml_algorithms["relative_percent"].to_list(),
    y_values=ml_algorithms["ML Algorithms"].to_list(),
    display_text=np.round((ml_algorithms["relative_percent"] *100), decimals = 2),
    top_n=3,
    rest_n=ml_algorithms.shape[0]-3,
    hovertext = ml_algorithms["counts"].to_list(),
    title="Top 12 Machine Learning Algorithms",
    subtitle="",
    orientation="h"
```

)

4.35% 5.59% 6.45% 7.11% 13.1% 16.99% 17.61% 18.94% 29.62% 32.82% 48.35% 56.81%

0.0%20.0%40.0%Evolutionary ApproachesGenerative Adversarial NetworksGraph Neural NetworksAutoencoder Networks (DAE, VAE, etc)Transformer Networks (BERT, gpt-3, etc)Recurrent Neural NetworksDense Neural Networks (MLPs, etc)Bayesian ApproachesConvolutional Neural NetworksGradient Boosting Machines (xgboost, lightgbm, etc)Decision Trees or Random ForestsLinear or Logistic Regression

Top 12 Machine Learning Algorithms

In terms of the top commonly used Machine Learning Algorithms we can see first in the list the **Linear or Logistic Regression**, followed by **Decision Trees or Random Forests**. That's neither a surprise for a couple of reasons:

1. These algorithms perform very well and achieve high accuracy in a variety of tasks with structured data,
2. they are easy to implement and they don't require huge hardware resources and time for training and/or inferencing.
3. Another important reason is that these Machine Learning methods offer **interpretability and explainability** that are becoming essential in solutions we build nowadays. Especially in fields such as healthcare or banking, interpretability and explainability could for example help overcome some legal constraints. **In solutions that support a human decision, it is essential to establish a trust relationship and explain the outcome and the internal mechanics of an algorithm. The whole idea behind interpretable and explainable ML is to avoid the black box effect.**

Next on the list is the **Gradient Boosting Machines** which are really powerful methods that usually achieve good accuracy, while later we can see the "Black Boxes algorithms" such as Convolutional Neural Networks, Transformer networks, Autoencoder, etc. that perform very well when we have unstructured data, such as text and images.

The same insights are also reflected in the second plot below, where it can be seen that Linear or Logistic Regression, and Decision Trees or Random Forests are commonly used across all sectors whereas Convolutional Neural Networks are most popular in tech companies, used by the **37%** of the respondents working in the tech sector. They are also used in the Academic field where research scientists explore new algorithms for processing images, videos or text. These sectors usually don't lack in training resources and interpretability is not a must-have.

unfold_moreShow hidden code

Linear or Logistic RegressionDecision Trees or Random ForestsGradient Boosting Machines Bayesian ApproachesEvolutionary ApproachesDense Neural Networks Convolutional Neural NetworksGenerative Adversarial NetworksRecurrent Neural NetworksTransformer Networks Autoencoder Networks Graph Neural NetworksAcademics/EducationAccounting/FinanceBroadcasting/CommunicationsComputers/Technology Energy/MiningGovernment/Public ServiceInsurance/Risk AssessmentManufacturing/FabricationMarketing/CRMMedical/PharmaceuticalNon-profit/ServiceOnline Service/Internet-based ServicesOtherRetail/SalesShipping/Transportation

102030405060PercentCommonly Used Machine Learning Algorithms in Different IndustriesQuestions Data: Industry (Q24) and ML Algorithm (Q18)Size,Color: Percentage of Respondents - The number of respondents of the related sector that chose the relevant ML Algorithmdivided by the total number of respondents working in that sector.

Transfer learning in the business world

Transfer learning is quite popular nowadays and it aims to save time and effort and provides the advantage of using tested models. This way, companies cut costs by avoiding the need for a high-cost GPU for retraining the model. The goal is to make machine learning as human as possible. Transfer learning is mostly used in **computer vision and natural language processing tasks** due to the huge amount of computational power required.

The following charts represent the percentage of respondents that use pre-trained models, specified below, for Computer Vision and NLP respectively on a regular basis.

It is clear that a higher percentage of respondents use pre-trained image classification models rather than transformer language models which is kinda expected due to **"ImageNet moment"**.

Pretraining entire models to learn both low and high-level features has been practiced for years by the computer vision (CV) community. Most often, this is done by learning to classify images on the large

ImageNet dataset. ULMFiT, ELMo, and the BERT model have the last years brought the NLP community an "ImageNet for language"---that is, a task that enables models to learn higher-level nuances of language, similarly to how ImageNet has enabled the training of CV models that learn general-purpose features of images. So, I expect the next years to see also a bigger percentage of professionals in AI use pre-trained models for NLP tasks.

unfold_lessHide code

In [25]:

```
map_cv_methods = {
    "Vision transformer networks (ViT, DeiT, BiT, BEiT, Swin, etc)": "Vision transformer<br>networks" ,
    "Generative Networks (GAN, VAE, etc)": "Generative Networks",
    "General purpose image/video tools (PIL, cv2, skimage, etc)": "General purpose<br><sup>image/video tools</sup>",
    "Object detection methods (YOLOv6, RetinaNet, etc)": "Object detection<br>methods",
    "Image classification and other general purpose networks (VGG, Inception, ResNet, ResNeXt, NASNet, EfficientNet, etc)": "Image classification Nets",
    "Image segmentation methods (U-Net, Mask R-CNN, etc)": "Image segmentation<br>methods"
}


map_nlp_methods = {
    "Contextualized embeddings (ELMo, CoVe)": "Contextualized<br>embeddings" ,
    "Encoder-decoder models (seq2seq, vanilla transformers)": "Encoder-decoder models",
    "Word embeddings/vectors (GLoVe, fastText, word2vec)": "Word embeddings<br><sup>GLoVe, fastText, word2vec</sup>",
    "Transformer language models (GPT-3, BERT, XLnet, etc)": "Transformer <br>language models",


}
```

```
computer_vision_methods =  extract_and_count_all_the_multiple_choice_answers("Q19", scope_df)

computer_vision_methods = computer_vision_methods[~computer_vision_methods["Q19"].isin(["None", "Other"])]

computer_vision_methods["Q19"] = computer_vision_methods["Q19"].apply(lambda x : map_cv_methods[x])



nlp_methods = extract_and_count_all_the_multiple_choice_answers("Q20", scope_df)

nlp_methods = nlp_methods[~nlp_methods["Q20"].isin(["None", "Other"])]

nlp_methods["Q20"] = nlp_methods["Q20"].apply(lambda x : map_nlp_methods[x])



pre_trained_models =  extract_and_count_all_the_multiple_choice_answers("Q21", scope_df)

pre_trained_models["Q21"] = np.where(pre_trained_models["Q21"] == "No, I do not download pre-trained model weights on a regular basis", "No, I do not download <br>pre-trained model weights", pre_trained_models["Q21"])



traces = dict()



# Creating the bar chart

trace_nlp = get_bar_plot_trace(

    nlp_methods["relative_percent"].to_list(),

    nlp_methods["Q20"].to_list(),

    np.round((nlp_methods["relative_percent"] *100), decimals = 2),
```

```
    2,
    nlp_methods.shape[0]-2,
    nlp_methods["counts"].to_list()
)


trace_cv = get_bar_plot_trace(
    computer_vision_methods["relative_percent"].to_list(),
    computer_vision_methods["Q19"].to_list(),
    np.round((computer_vision_methods["relative_percent"] *100), decimals = 2),
    2,
    computer_vision_methods.shape[0]-2,
    computer_vision_methods["counts"].to_list()
)


trace_models = get_bar_plot_trace(
    pre_trained_models["Q21"].apply(lambda x : x.split("(")[0]).to_list(),
    pre_trained_models["relative_percent"].to_list(),
    np.round((pre_trained_models["relative_percent"] *100), decimals = 2),
    3,
    pre_trained_models.shape[0]-3,
    pre_trained_models["counts"].to_list(),
    orientation = "v"
```

```
)

traces["NLP_methods"] = trace_nlp
traces["CV_methods"] = trace_cv


fig = make_subplots(
    rows=1,
    cols=2 ,
    shared_yaxes=False,
    shared_xaxes=True,
    horizontal_spacing = 0.15,
    subplot_titles=("Most common Computer Vision methods", "Most common NLP methods", "Do you download Pre-Trained Models for Transfer Learning?"))


fig.append_trace(traces["CV_methods"],1,1)
fig.append_trace(traces["NLP_methods"],1,2)

large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>How Transfer Learning is being used today</span>"
small_title_format = "<span style='font-size:14px; font-family:Helvetica'>The length of the bars denotes the <b>percentage of professionals in the field that use the specified model</b>.</span>"


layout = dict(
```

```
    title = large_title_format + "<br>" + small_title_format + "<br><br>",

    showlegend = False,

    font = dict(color = '#7b6b59'),

    margin = dict(t=150),

    plot_bgcolor='#fff',

    bargap = 0.10,


)



fig['layout'].update(layout)


fig.show()


large_title_format = "<span style='font-size:22px; font-family:Times New Roman'>Do you download pre-trained model weights from any <br>of the public available services? </span>"



fig = go.Figure(trace_models)

layout = dict(

    title = large_title_format + "<br>",

    showlegend = False,

    font = dict(color = '#7b6b59'),

    margin = dict(t=40),

    plot_bgcolor='#fff',
```

```
    bargap = 0.10,

)



fig['layout'].update(layout)


fig.show()
```

4.2% 6.58% 12.25% 12.27% 13.21% 18.76% 00.050.10.15Vision transformernetworksGenerative NetworksGeneral purposeimage/video toolsImage segmentationmethodsObject detectionmethodsImage classification Nets3.68% 9.16% 13.18% 13.2% 00.050.1ContextualizedembeddingsEncoder-decoder modelsWord embeddingsGLoVe, fastText, word2vecTransformer language models

How Transfer Learning is being used todayThe length of the bars denotes the percentage of professionals in the field that use the specified model.Most common Computer Vision methodsMost common NLP methods

0.89% 2.54% 2.76% 3.12% 3.24% 10.51% 11.93% 15.74% 23.67% 37.49% Jumpstart ONNX models Timm Other storage services NVIDIA NGC models PyTorch Hub Huggingface Models TensorFlow Hub Kaggle datasets No, I do not download pre-trained model weights00.050.10.150.20.250.30.35

Do you download pre-trained model weights from any of the public available services?

NLP Users

In the tables below, we can then see the number of professionals that use pre-trained models and methods for NLP / CV tasks on a regular basis along with the relative percentages. The percentages column has been calculated by dividing the number of professionals in each role that use CV/NLP methods by the total number of respondents that have this job role. The key takeaway is that CV / NLP methods and pre-trained models are used mostly by **Machine Learning Engineers**, **Data Scientists**, **Data Architects**, **Developer Advocate**, and **Research Scientists**.

unfold_moreShow hidden code

Out[26]:

	Use of NLP Methods and Pre-trained Models	Nbr of respondents	%
0	No	7399	81.360000
1	Yes	1695	18.640000

unfold_lessHide code

In [27]:

```
# Get the counts of occurrences of each job role
roles_totals = scope_df["Q23"].value_counts().to_dict()


nlp_usage = scope_df[scope_df["NLP_methods_usage"] == "Yes"].groupby(["Q23"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "Nbr of respondents", "Q23" : "Role"})


nlp_usage["%"] = nlp_usage.apply(lambda x : x["Nbr of respondents"] / roles_totals[x["Role"]], axis = 1)
nlp_usage["%"]  = np.round(nlp_usage["%"] * 100, 2)
nlp_usage = nlp_usage.sort_values(by=["%"], ascending=False).reset_index(drop=True)


nlp_usage.style.background_gradient(axis=0,  cmap='Oranges')
```

Out[27]:

	Role	Nbr of respondents	%
0	Machine Learning/ MLops Engineer	251	44.660000
1	Data Scientist	582	30.420000
2	Developer Advocate	17	28.810000
3	Research Scientist	143	24.240000
4	Data Architect	20	21.050000
5	Data Engineer	57	16.720000
6	Software Engineer	157	16.170000
7	Manager (Program, Project, Operations, Executive-level, etc)	132	15.980000
8	Teacher / professor	120	14.630000

9	Statistician	12	9.760000
10	Data Analyst (Business, Marketing, Financial, Quantitative, etc)	116	7.670000
11	Data Administrator	5	7.140000
12	Engineer (non-software)	32	6.910000
13	Other	51	6.820000

Computer Vision Users

unfold_lessHide code

In [28]:

```
cv_usage = scope_df.groupby(
    ["CV_methods_usage"]
).agg({
    "Q2" : "count"
}).reset_index().rename(columns={
    "Q2": "Nbr of respondents",
    "CV_methods_usage": "Use of CV Methods and Pre-trained Models"
})
cv_usage["%"] = np.round((cv_usage["Nbr of respondents"] / scope_df.shape[0]) * 100, 2)
```

```
cv_usage.style.background_gradient(axis=0,  cmap='Blues')
```

Out[28]:

	Use of CV Methods and Pre-trained Models	Nbr of respondents	%
0	No	6705	73.730000
1	Yes	2389	26.270000

unfold_lessHide code

In [29]:

```
# Get the counts of occurrences of each job role

roles_totals = scope_df["Q23"].value_counts().to_dict()



cv_usage = scope_df[scope_df["CV_methods_usage"] == "Yes"].groupby(["Q23"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "Nbr of respondents", "Q23" : "Role"})



cv_usage["%"] = cv_usage.apply(lambda x : x["Nbr of respondents"] / roles_totals[x["Role"]], axis = 1)

cv_usage["%"]  = np.round(cv_usage["%"] * 100, 2)

cv_usage = cv_usage.sort_values(by=["%"], ascending=False).reset_index(drop=True)



cv_usage.style.background_gradient(axis=0,  cmap='Oranges')
```

Out[29]:

	Role	Nbr of respondents	%
0	Machine Learning/ MLops Engineer	339	60.320000
1	Research Scientist	241	40.850000
2	Developer Advocate	22	37.290000
3	Data Scientist	613	32.040000
4	Data Architect	30	31.580000
5	Teacher / professor	242	29.510000
6	Software Engineer	283	29.150000
7	Data Engineer	90	26.390000
8	Manager (Program, Project, Operations, Executive-level, etc)	194	23.490000

9	Engineer (non-software)	73	15.770000
10	Other	90	12.030000
11	Data Analyst (Business, Marketing, Financial, Quantitative, etc)	155	10.240000
12	Data Administrator	7	10.000000
13	Statistician	10	8.130000

Usage of specialized hardware for ML models training

There are broadly 2 stages to a Machine Learning project. The first stage is ML **Model Training** and the second stage is the **Model Inference**.

Training an ML model requires more computational power and resource. Especially when working with Neural Networks, it is essential to process huge amounts of data to train the model. This process usually involves some heavy matrix calculations. GPUs are a specialized hardware used for Machine Learning because they can perform multiple, simultaneous computations. This enables the distribution of training processes and can significantly speed up machine learning operations. With GPUs, we can accumulate many cores that use fewer resources without sacrificing efficiency or power. However, GPU is not the only specialized hardware that is used for ML. There are also other types of specialized hardware as we'll see below, but the GPU is the one that is used most commonly.

So, when designing our deep learning architecture we have to consider multiple factors for our decision to use GPUs or any other specialized hardware or not (dataset size, model size, etc.). **As the survey data shows only 31% of the respondents use specialized hardware like GPU for ML model training.**

unfold_lessHide code

In [30]:

```
hardware_usage = scope_df.groupby(
    ["GPU_usage"]
).agg({
    "Q2" : "count"
}).reset_index().rename(columns={
    "Q2": "Nbr of respondents",
    "GPU_usage": "Specialized Hardware Usage"
})
hardware_usage["%"] = np.round((hardware_usage["Nbr of respondents"] / scope_df.shape[0]) * 100, 2)
hardware_usage.style.background_gradient(axis=0, cmap='Blues')
```

Out[30]:

	Specialized Hardware Usage	Nbr of respondents	%
0	No	6263	68.870000
1	Yes	2831	31.130000

unfold_moreShow hidden code

0. Not Started(No ML)1. ExplorationOnly Exploring ML2. Beginner StageUse ML only for Insights3. Intermediate StageRecently Started Using ML4. Advance StageWell Established MLNot Known0100200300400500600700800

Specialized hardware usage: YesSpecialized hardware usage for ML models training by ML adoption stage

Companies with Machine Learning Models in production either in an advanced or intermediate stage are more likely than the ones that started recently exploring ML capabilities to use GPUs for training their ML Models as it can be seen in the illustration above.

unfold_lessHide code

In [32]:

```
dfs_list = []


for col in [column for column in df.columns if column.startswith("Q42")]:

    dfs_list.append(scope_df.groupby([col]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts", col: "Hardware"}))


hardware = pd.concat(dfs_list)

hardware["relative_percent"] = hardware.apply(lambda x : x["counts"] / scope_df.shape[0], axis = 1)



hardware = hardware.sort_values(by=["relative_percent"], ascending=True)

hardware = hardware[~hardware["Hardware"].isin(["None", "Other"])]


create_single_bar_plot(

    x_values=hardware["relative_percent"].to_list(),
```

```
    y_values=hardware["Hardware"].to_list(),

    display_text=np.round((hardware["relative_percent"] *100), decimals = 2),

    top_n=2,

    rest_n=hardware.shape[0]-2,

    hovertext = hardware["counts"].to_list(),

    title="Commonly Used Types of Specialized Hardware",

    subtitle="",

    orientation="h"

)
```

0.29% 0.43% 0.64% 0.64% 0.74% 7.18% 29.49% 0.0%5.0%10.0%15.0%20.0%25.0%30.0% WSEs Trainium Chips RDUs Inferentia Chips IPUs TPUs GPUs

Commonly Used Types of Specialized Hardware

Specialized Hardware Users

The table below shows the number of professionals that use specialized hardware for ML model training. The percentages column has been calculated by dividing the number of professionals in each role that use GPUs or TPUs, etc. by the total number of respondents that have the same job role.

unfold_lessHide code

In [33]:

```
roles_totals = scope_df["Q23"].value_counts().to_dict()


gpu_usage = scope_df[scope_df["GPU_usage"] == "Yes"].groupby(["Q23"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "Nbr of respondents", "Q23" : "Role"})
```

```
gpu_usage["%"] = gpu_usage.apply(lambda x : x["Nbr of respondents"] / roles_totals[x["Role"]], axis = 1)

gpu_usage["%"]  = np.round(gpu_usage["%"] * 100, 2)

gpu_usage = gpu_usage.sort_values(by=["%"], ascending=False).reset_index(drop=True)


gpu_usage.style.background_gradient(axis=0,  cmap='Oranges')
```

Out[33]:

	Role	Nbr of respondents	%
0	Machine Learning/ MLops Engineer	352	62.630000
1	Data Scientist	811	42.390000
2	Research Scientist	242	41.020000
3	Data Engineer	119	34.900000
4	Data Architect	33	34.740000
5	Manager (Program, Project, Operations, Executive-level, etc)	277	33.540000

6	Software Engineer	291	29.970000
7	Developer Advocate	16	27.120000
8	Teacher / professor	203	24.760000
9	Engineer (non-software)	85	18.360000
10	Data Analyst (Business, Marketing, Financial, Quantitative, etc)	264	17.450000
11	Other	112	14.970000
12	Data Administrator	10	14.290000
13	Statistician	16	13.010000

AI job roles and key skills needed to build a career in AI

Whether the insights from the 2022 Kaggle Machine Learning & Data Science Survey illustrated in this notebook so far or the progress Artificial Intelligence and Machine Learning has made today excite you to get into the AI and Data Science world and build a career in AI, this section is the right place for you 😁!!! In this part, I'll provide some insights about the different job roles and the top skills required, based on the responses of the professionals who participated in the survey.


PASSION LED US HERE

Photo by Ian Schneider on Unsplash

As it has been seen in the above sections, a lot of companies across different industries are adopting AI solutions. Enterprises have also recognized the benefits of having an in-house team for data analytics. This has led to the rise of AI-related jobs. However, the different titles present in the market may confuse a newcomer. Different titles also require different specializations, which makes it difficult for an aspirant to choose the role they are equipped for and interested in.

AI jobs description: roles, responsibilities and skills required

So let's have first a look at the most in-demand AI jobs according to the survey respondents that already have a job position related to AI.

unfold_lessHide code

In [34]:

```
data_science_roles = scope_df.groupby(["Q23"]).agg({"Q2" :
"count"}).reset_index().rename(columns={"Q2": "counts"})


data_science_roles["relative_percent"] = data_science_roles.apply(lambda x : (x["counts"] /
scope_df.shape[0]), axis = 1)

data_science_roles = data_science_roles.sort_values(by=["relative_percent"], ascending=True)

data_science_roles = data_science_roles[~data_science_roles["Q23"].isin(["None", "Other"])]



create_single_bar_plot(

    x_values=data_science_roles["relative_percent"].to_list(),

    y_values=data_science_roles["Q23"].to_list(),

    display_text=np.round((data_science_roles["relative_percent"] *100), decimals = 2),

    top_n=2,
```

```
    rest_n=data_science_roles.shape[0]-2,

    hovertext = data_science_roles["counts"].to_list(),

    title="Top AI Jobs in the Market",

    subtitle="",

    orientation="h"

)
```

0.65% 0.77% 1.04% 1.35% 3.75% 5.09% 6.18% 6.49% 9.02% 9.08% 10.68% 16.64% 21.04% 0.0%5.0%10.0%15.0%20.0%Developer AdvocateData AdministratorData ArchitectStatisticianData EngineerEngineer (non-software)Machine Learning/ MLops EngineerResearch ScientistTeacher / professorManager (Program, Project, Operations, Executive-level, etc)Software EngineerData Analyst (Business, Marketing, Financial, Quantitative, etc)Data Scientist

Top AI Jobs in the Market

Unsurprisingly, the **Data Scientists** ranked first in the chart with the most common data-related jobs. With 1,913 respondents they form 21.04% of our data professionals (9,094 in total), considerably ahead of **Data Analysts** in second place with 16.64%, followed by **Software Engineers** with 10.68%.

But what industries are actually hiring AI specialists and what AI roles do they seek??

unfold_lessHide code

In [35]:

```
roles_df = scope_df.groupby(["Q24", "Q23"]).agg({"Q2" :
"count"}).reset_index().rename(columns={"Q2": "counts"})

roles_df["relative_percent"] = roles_df.apply(lambda x : x["counts"] / industry_totals[x["Q24"]], axis = 1)


create_scatter_plot(

    roles_df["Q23"].apply(lambda x : x.split("(")[0]),
```

```
roles_df["Q24"],

 "Role: %{x}<br>" +

   "Industry: %{y}<br>" +

   "Percentage: %{marker.size:,}" +

   "<extra></extra>",

roles_df['relative_percent']*100,

roles_df['relative_percent']*100,

"What Industries are Hiring the Most AI Technology Specialists?",

"Questions Data: Industry (Q24) and Job Role (Q23)",

"Size,Color: Percentage of Respondents - <br>The number of respondents with the relevant job position in the related sector<br>divided by the total number of respondents working in that sector."

)
```

Data AdministratorData Analyst Data ArchitectData EngineerData ScientistDeveloper AdvocateEngineer Machine Learning/ MLops EngineerManager OtherResearch ScientistSoftware EngineerStatisticianTeacher / professorAcademics/EducationAccounting/FinanceBroadcasting/CommunicationsComputers/Technology Energy/MiningGovernment/Public ServiceInsurance/Risk AssessmentManufacturing/FabricationMarketing/CRMMedical/PharmaceuticalNon-profit/ServiceOnline Service/Internet-based ServicesOtherRetail/SalesShipping/Transportation

51015202530354045PercentWhat Industries are Hiring the Most AI Technology Specialists?Questions Data: Industry (Q24) and Job Role (Q23)Size,Color: Percentage of Respondents - The number of respondents with the relevant job position in the related sectordivided by the total number of respondents working in that sector.

The scatter plot shows that 37.10% of employees in **Insurance companies** are **Data Scientists**, making them top the list of industries hiring Data Scientists. Data science can enable insurers to develop effective strategies to acquire new customers, develop personalized products, analyze risks, assist underwriters, implement fraud detection systems, and much more.

Second in the list with the sectors that occupy the most data scientists proportionally with the total number of respondents working in that sector is the **Marketing** and **CRM** companies, followed by the **Retail/Sales** field and the companies offering **Internet-based services**. A wider range of information is available to these companies, therefore Data science helps them to put these data to efficient use to drive more business and refine their products/services offerings. These sectors as it can be seen also seek Data Analysts.

Now, let's focus on the **Data Scientists** and **Data Analysts** since they are the most popular job roles as well as on the **Machine Learning Engineers** and **Research Scientists** who are core components of the AI & Data Science teams, and see how a typical day at work looks like. Let's see the main tasks and the responsibilities that they have.

Note: In order to create the following chart, for each activity, I counted the number of respondents (Data Scientists, Analysts, ML engineers) who chose it and I calculated the percentages of each activity that you see below based on their total sum.

unfold_lessHide code

In [36]:

```
dfs_list = []
ml_scope_df = scope_df[
    (scope_df["Q23"].isin(["Machine Learning/ MLops Engineer", "Data Scientist"])) |
    (scope_df["Q23"].str.contains("Data Analyst"))
]

for col in [column for column in df.columns if column.startswith("Q28")]:
    dfs_list.append(ml_scope_df.groupby([col]).agg({"Q2" :
"count"}).reset_index().rename(columns={"Q2": "counts", col: "ML Activities"}))

ml_activities = pd.concat(dfs_list)
```

```
ml_activities["relative_percent"] = ml_activities.apply(lambda x : x["counts"] /
ml_activities["counts"].sum(), axis = 1)



ml_activities = ml_activities.sort_values(by=["relative_percent"], ascending=False)

ml_activities = ml_activities[

    ~((ml_activities["ML Activities"].str.contains("None")) |

    (ml_activities["ML Activities"].str.contains("Other")))

]



map_ml_activities = {

    "Analyze and understand data to influence product or business decisions": "Analyze and understand
data<br>to influence product or business decisions" ,

    "Build prototypes to explore applying machine learning to new areas": "Build prototypes to explore
<br>applying machine learning to new areas",

    "Build and/or run the data infrastructure that my business uses for storing, analyzing, and
operationalizing data": "Build and/or run the data infrastructure",

    "Experimentation and iteration to improve existing ML models": "Experimentation and iteration<br>to
improve existing ML models",

    "Build and/or run a machine learning service that operationally improves my product or workflows":
"Build and/or run a machine learning service",

    "Do research that advances the state of the art of machine learning": "Do research that advances
the<br>state of the art of machine learning"



}

ml_activities["ML Activities"] = ml_activities["ML Activities"].apply(lambda x : map_ml_activities[x])
```

```
fig = go.Figure(go.Funnelarea(

      values = ml_activities["counts"].to_list(), text = ml_activities["ML Activities"].to_list(),

      marker = {"colors": ["#a43725","#c07156", "#E6b6a4", "#edc860", "#e5b01c", "#cfbd9b", "#a43725"],

              },

      textfont = {"family": "Times New Roman", "size": 22, "color": "black"}, opacity = 0.65))


large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>A Day in the Life of a Data Scientist / Analyst or ML Engineer</span>"



layout = dict(

    title = large_title_format,

    font = dict(color = '#7b6b59'),

    margin = dict(t=170),

    width = 800,

    height= 700,

    plot_bgcolor = "white"

)


fig.update_layout(layout)

fig.update_traces(showlegend=False)

fig.show()
```

Analyze and understand datato influence product or business decisions29.8%Build prototypes to explore applying machine learning to new areas18%Build and/or run the data infrastructure15.3%Experimentation and iterationto improve existing ML models14.9%Build and/or run a machine learning service14%Do research that advances thestate of the art of machine learning7.93%

A Day in the Life of a Data Scientist / Analyst or ML Engineer

The top level of the reversed pyramid represents the most common activity whereas going down we see the tasks, implemented less commonly. In addition to that, you can also see the most relevant activities per role in the illustrations below.

Key insights:

- So, **29.8%** of the total activities that the respondents do is **Analyze and understand data to influence product or business decisions.** Data analysis dominates Data Scientists and Data Analysts' activities as is also illustrated in the following visualizations. The main task of those two roles is to analyze data to identify patterns and trends and extracts actionable insights for driving business decisions.
- The second most common activity is to **implement Machine Learning methods to explore new areas**. In this task Machine Learning Engineers, Data Scientists, and Research Scientists are mainly involved.
- In the third and fourth positions are the **Experimentation and iteration to improve existing ML models** and **Build a machine learning service**. Perhaps is not a surprise that Machine Learning Engineers are mainly responsible for these activities.
- One less common activity is to **Build and run data infrastructure** where all 4 roles contribute almost equally.
- Last but not least, is to **Do research that advances the state of the art of machine learning** which as it's expected undertaken mostly by Research Scientists.

unfold_lessHide code

In [37]:

```
jobs_in_scope = [
    "Data Scientist",
    "Data Analyst (Business, Marketing, Financial, Quantitative, etc)",
    "Research Scientist",
```

```
    "Machine Learning/ MLops Engineer"
]
activities = [col for col in df.columns if col.startswith("Q28")]
job_roles = scope_df["Q23"].str.strip().value_counts().to_dict()
dfs_list = []
for role in jobs_in_scope:
    for col in activities:
        roles_df = scope_df[
            scope_df["Q23"].str.strip() == role
        ].groupby(["Q23", col]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts", col: "ML Activities"})
        dfs_list.append(roles_df)


results = pd.concat(dfs_list)


results["Q23"] = results["Q23"].str.strip()
results["relative_percent"] = results.apply(lambda x : x["counts"] / job_roles[x["Q23"]], axis = 1)
results = results[
    ~((results["ML Activities"].str.contains("None")) |
    (results["ML Activities"].str.contains("Other")))
]



map_ml_activities = {
```

```
    "Analyze and understand data to influence product or business decisions": "1. Analyze and understand data<br><sup>to influence product or business decisions</sup>" ,

    "Build prototypes to explore applying machine learning to new areas": "2. Build prototypes to explore <br><sup>applying machine learning to new areas</sup>",

    "Build and/or run the data infrastructure that my business uses for storing, analyzing, and operationalizing data": "3. Build and/or run the data infrastructure</sup>",

    "Experimentation and iteration to improve existing ML models": "4. Experimentation and iteration<br><sup>to improve existing ML models</sup>",

    "Build and/or run a machine learning service that operationally improves my product or workflows": "5. Build and/or run a machine learning service",

    "Do research that advances the state of the art of machine learning": "6. Do research that advances <br><sup>the state of the art of machine learning</sup>"


}

results["ML Activities"] = results["ML Activities"].apply(lambda x : map_ml_activities[x])



results = results.sort_values(by=["ML Activities"], ascending=False)




create_scatter_plot(

    results["Q23"].apply(lambda x : x.split("(")[0]).to_list(),

    results["ML Activities"].apply(lambda x : x.split("(")[0]),

     "Role: %{x}<br>" +

       "ML Activity: %{y}<br>" +

       "Percentage: %{marker.size:,}" +

       "<extra></extra>",

    results['relative_percent']*100,
```

```
    results['relative_percent']*100,
    "Tasks among ML and Data Science Roles",
    "Questions Data: ML Activity (Q28) and Job Role (Q23)",
    "Size,Color: Percentage of Respondents - <br>The number of respondents with the relevant job position doing the respective ML activity<br>divided by the total number of respondents with the same job position."
)
```

Machine Learning/ MLops EngineerResearch ScientistData ScientistData Analyst 6. Do research that advances the state of the art of machine learning5. Build and/or run a machine learning service4. Experimentation and iterationto improve existing ML models3. Build and/or run the data infrastructure2. Build prototypes to explore applying machine learning to new areas1. Analyze and understand datato influence product or business decisions

10203040506070PercentTasks among ML and Data Science RolesQuestions Data: ML Activity (Q28) and Job Role (Q23)Size,Color: Percentage of Respondents - The number of respondents with the relevant job position doing the respective ML activitydivided by the total number of respondents with the same job position.

unfold_lessHide code

In [38]:

```
jobs_in_scope = [
    "Data Scientist",
    "Data Analyst (Business, Marketing, Financial, Quantitative, etc)",
    "Research Scientist",
    "Machine Learning/ MLops Engineer"
]
```

```
tasks_in_scope = [
    "Q28_1",
    "Q28_2",
    "Q28_3",
    "Q28_4",
    "Q28_5",
    "Q28_6",
]

label = [
    "Data Scientist", #0
    "Data Analyst", #1
    "Research Scientist", #2
    "Machine Learning Engineer", #3
    'Analyze and Understand Data', #4
    'Build and run data infrastructure', #5
    'Create ML to explore new areas', #6
    'Build and run ML',  #7
    'Improve ML Models', #8
    'Research to advance the state of ML' #9

]

source = [0,0,0,0,0,0, 1,1,1,1,1,1,   2,2,2,2,2,2, 3,3,3,3,3,3]
```

```
target = [4,5,6,7,8,9,  4,5,6,7,8,9, 4,5,6,7,8,9, 4,5,6,7,8,9,]

value = []

for job in jobs_in_scope:

    for col in tasks_in_scope:



        value.append(scope_df[scope_df["Q23"] == job ][col].count())



# Colors

color_node = [

    "#CC5600",

    "#9D4800",

    "#91281A",

    "#DA9300",

    "#325C6E",

    "#325C6E",

    "#325C6E",

    "#325C6E",

    "#325C6E",

    "#325C6E",

    "#325C6E"

]



color_link = ["#F8E8DC","#CC5600", "#F8E8DC", "#F8E8DC", "#CC5600",

              "#EBD5C3","#EBD5C3", "#9D4800", "#EBD5C3", "#9D4800",
```

```
        "#DDCECC", "#DDCECC", "#91281A", "#DDCECC", "#91281A",
        "#F8EED9", "#DA9300", "#F8EED9", "#F8EED9", "#DA9300"]


color_link = ["#CC5600", "#F8E8DC", "#CC5600", "#F8E8DC", "#F8E8DC", "#F8E8DC",
        "#9D4800", "#9D4800", "#EBD5C3", "#EBD5C3","#EBD5C3","#EBD5C3",
        "#91281A", "#DDCECC", "#91281A", "#DDCECC", "#DDCECC", "#DDCECC",
        "#F8EED9", "#F8EED9", "#DA9300", "#F8EED9", "#DA9300", "#F8EED9",
       ]

fig = go.Figure(data=[go.Sankey(
  node = dict(
   pad = 10,
   thickness = 21,
   line = dict(color = "black", width = 0.5),
   label = label,
   color=color_node,


  ),
  link = dict(
   source = source, # indices correspond to labels, eg A1, A2, A1, B1, ...
   target = target,
   value = value,
```

```
        color = color_link

  ), arrangement='snap')])


# title format

large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Tasks among ML and Data Science Roles</span>"


layout = dict(

    #title = large_title_format,

    font = dict(color = '#7b6b59'),


)


fig.update_layout(layout)

fig.show()
```

Data ScientistData AnalystResearch ScientistMachine Learning EngineerAnalyze and Understand DataBuild and run data infrastructureCreate ML to explore new areasBuild and run MLImprove ML ModelsResearch to advance the state of ML

unfold_lessHide code

In [39]:

```
jobs_in_scope = [

    "Data Scientist",

    "Data Analyst (Business, Marketing, Financial, Quantitative, etc)",
```

```
    "Research Scientist",

    "Machine Learning/ MLops Engineer"

]


models_in_scope = [

    "Models in Production",

    "Not Started",

    "Exploration Stage",

    "Generating Insights"


]

tasks_in_scope = [

    "Q28_1",

    "Q28_2",

    "Q28_3",

    "Q28_4",

    "Q28_5",

    "Q28_6",

]


label = [

    "Data Scientist", #0

    "Data Analyst", #1

    "Research Scientist", #2
```

```
    "Machine Learning Engineer", #3

    "Models in Production", #4

    "Not Started", #5

    "Exploration Stage", #6

    "Generating Insights", #7

    'Analyze and Understand Data', #8

    'Build and run data infrastructure', #9

    'Create ML to explore new areas', #10

    'Build and run ML', #11

    'Improve ML Models', #12

    'Research to advance the state of ML' #13


]

source = [0, 0, 0, 0, 4,4,4,4,4,4, 5,5,5,5,5,5, 6,6,6,6,6,6, 7,7,7,7,7,7,

        1, 1, 1, 1, 4,4,4,4,4,4, 5,5,5,5,5,5, 6,6,6,6,6,6, 7,7,7,7,7,7,

          2, 2, 2, 2, 4,4,4,4,4,4, 5,5,5,5,5,5, 6,6,6,6,6,6, 7,7,7,7,7,7,

         3,3,3,3, 4,4,4,4,4,4, 5,5,5,5,5,5, 6,6,6,6,6,6, 7,7,7,7,7,7,

        ]

target = [4, 5, 6, 7,  8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13,

         4, 5, 6, 7,  8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13,

         4, 5, 6, 7,  8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13,

         4, 5, 6, 7,  8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13, 8,9,10,11,12,13, ]

value = []

for job in jobs_in_scope:
```

```
for model in models_in_scope:
    value.append(
        scope_df[
            (scope_df["Q23"] == job) &
            (scope_df["ML_adoption_class"] == model)
        ].shape[0])
for model in models_in_scope:
    for col in tasks_in_scope:

        value.append(
          scope_df[
           (scope_df["Q23"] == job) &
           (scope_df["ML_adoption_class"] == model)
        ][col].count())




# Colors
color_node = ["#CC5600", "#9D4800",  "#91281A", "#DA9300"] + ["#c07156"]*4 + ["#325C6E"]*6



color_link = ["#DDCECC"]*4 + ["#89CFF0"]*24 +["#DA9300"]*4 +["pink"]*24 + ["#FAC898"] * 4 +
["pink"]*24 + ["#F8EED9"] * 4 + ["pink"]*24


fig = go.Figure(data=[go.Sankey(
```

```
    node = dict(
      pad = 15,
      thickness = 20,
      line = dict(color = "black", width = 0.5),
      label = label,
     color=color_node,



    ),
    link = dict(
      source = source, # indices correspond to labels, eg A1, A2, A1, B1, ...
      target = target,
      value = value,
       # color = color_link
  ))])


# title format
large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Tasks among ML and Data Science Roles</span>"


layout = dict(


    font = dict(color = '#7b6b59'),
```

```
)

fig.update_layout(layout)
fig.show()
```

Data ScientistData AnalystResearch ScientistMachine Learning EngineerModels in ProductionNot StartedExploration StageGenerating InsightsAnalyze and Understand DataBuild and run data infrastructureCreate ML to explore new areasBuild and run MLImprove ML ModelsResearch to advance the state of ML

unfold_lessHide code

In [40]:

```
years_ml_in_scope = list(map_ml_experience.values())[0:-1]
years_ml_in_scope = years_ml_in_scope[0:-1]

ml_activities = [col for col in scope_df.columns if col.startswith("Q28")]
# Exclude None and others
ml_activities = ml_activities[:-2]

ml_activities.reverse()
x = years_ml_in_scope
y = ['Do research that advances <br> the state of the art of machine learning',
 'Experimentation and iteration<br> to improve existing ML models',
 'Build and/or run a machine learning <br>service that operationally improves my product or workflows',
 'Build prototypes to explore <br>applying machine learning to new areas',
```

```
 'Build and/or run the data infrastructure that my<br> business uses for storing, analyzing, and operationalizing data',

 'Analyze and understand data to <br>influence product or business decisions']



z = []



for activity in ml_activities:

    tmp = []

    for years in years_ml_in_scope:

        tmp.append(round((scope_df[scope_df["Q16"] == years][activity].count() / scope_df[scope_df["Q16"] == years].shape[0]),2))

    z.append(tmp)



create_heatmap(z, x, y, z, "YlOrBr", "ML Experience in different responsibilities", subtitle="This helps us understand the level of ML experience needed to perform an activity.")
```

1. 0 years2. < 1 years3. 1-2 years4. 2-3 years5. 3-4 years6. 4-5 years7. 5-10 years8. 10-20 yearsDo research that advances the state of the art of machine learning Experimentation and iteration to improve existing ML models Build and/or run a machine learning service that operationally improves my product or workflows Build prototypes to explore applying machine learning to new areas Build and/or run the data infrastructure that my business uses for storing, analyzing, and operationalizing data Analyze and understand data to influence product or business decisions

ML Experience in different responsibilitiesThis helps us understand the level of ML experience needed to perform an activity.0.050.110.190.220.270.290.320.430.040.120.230.310.440.470.560.570.050.110.250.310.40.410.470.420.070.170.330.460.530.580.670.650.230.220.330.350.340.370.390.360.50.480.570.590.610.610.650.63

The chart above shows the percentage of respondents at a particular Machine Learning experience level for each responsibility. This helps us understand the level of ML expertise needed to perform a task.

The main key takeaways are:

- Data Analysis activities show higher percentages of individuals with ML experience of 2-3 years or more.
- Machine learning-related tasks such as Applying ML methods to new areas and improving existing ML models have greater percentages at the higher experience ranges.

Below you can see the distribution of the years of coding experience and experience using ML methods. While a big group of respondents has many years of coding they don't have many years experience in using Machine Learning methods.

unfold_lessHide code

In [41]:

```
programming_experience_df = scope_df.groupby(["Q11"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts"})

programming_experience_df["relative_percent"] = programming_experience_df.apply(lambda x : x["counts"] / scope_df.shape[0], axis = 1)

programming_experience_df = programming_experience_df.sort_values(by=["Q11"])

programming_experience_df["Q11"] = programming_experience_df["Q11"].apply(lambda x : x.split(".")[-1])



ml_experience_df = scope_df.groupby(["Q16"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts"})

ml_experience_df["relative_percent"] = ml_experience_df.apply(lambda x : x["counts"] / scope_df.shape[0], axis = 1)

ml_experience_df = ml_experience_df.sort_values(by=["Q16"])

ml_experience_df["Q16"] = ml_experience_df["Q16"].apply(lambda x : x.split(".")[-1])
```

```
traces = dict()

# Creating the bar chart
trace_experience_coding = get_bar_plot_trace(
    programming_experience_df["Q11"].to_list(),
    programming_experience_df["relative_percent"].to_list(),
    np.round((programming_experience_df["relative_percent"] *100), decimals = 2),
    0,
    programming_experience_df.shape[0]-0,
    programming_experience_df["counts"].to_list(),
    orientation="v"
)

trace_experience_ml = get_bar_plot_trace(
    ml_experience_df["Q16"].apply(lambda x : x.split("(")[0]),
    ml_experience_df["relative_percent"].to_list(),
    np.round((ml_experience_df["relative_percent"] *100), decimals = 2),
    0,
    ml_experience_df.shape[0]-0,
    ml_experience_df["counts"].to_list(),
    orientation="v"
)
```

```
fig = make_subplots(
    rows=1,
    cols=2 ,
    shared_yaxes=False,
    shared_xaxes=True,
    horizontal_spacing = 0.20,
    vertical_spacing = 0.10,
    subplot_titles=("Years of Coding Experience", "Years of using ML Methods")
)
traces["Programming_Experience"] = trace_experience_coding
traces["ML_experience"] = trace_experience_ml



fig.append_trace(traces["Programming_Experience"],1,1)
fig.append_trace(traces["ML_experience"],1,2)


large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Professional subgroups</span>"
small_title_format = "<span style='font-size:14px; font-family:Helvetica'>Python Is Essential for Data Analysis and Data Science.</b></span>"
```

```
layout = dict(
    title = large_title_format + "<br>" + small_title_format,
    font = dict(color = '#7b6b59'),
    showlegend = False,
    margin = dict(t=150,pad=6),
    plot_bgcolor='#fff',
    bargap = 0.10,
)

fig['layout'].update(layout)

fig.show()

programming_experience = list(map_programming_experience.values())[1:-1]
programming_experience.reverse()
ml_experience = list(map_ml_experience.values())[0:-1]

z = []
z_text = []


for coding in programming_experience:
    tmp = []
```

```
    tmp_text = []
    for ml in ml_experience:


        tmp.append((scope_df[(scope_df["Q16"] == ml) & (scope_df["Q11"] == coding)].shape[0]))
        num = (scope_df[(scope_df["Q16"] == ml) & (scope_df["Q11"] == coding)].shape[0])
        if coding in ["2. < 1 years" , "3. 1-3 years"] and ml in ["1. 0 years", "2. < 1 years"]:
            tmp_text.append(f"<b>Begginers</b><br><br>{num}")
        elif coding in ["4. 3-5 years" , "5. 5-10 years"] and ml in ["2. < 1 years", "3. 1-2 years", "4. 2-3 years",]:
            tmp_text.append(f"<b>Mid Level</b><br><br>{num}")
        elif coding in ["6. 10-20 years" , "7. 20+ years"] and ml in ["1. 0 years", "2. < 1 years",]:
            tmp_text.append(f"In<br>Transition<br>{num}")
        elif coding in ["6. 10-20 years" , "7. 20+ years"] and ml in ["7. 5-10 years", "8. 10-20 years",]:
            tmp_text.append(f"<b>ML Experts</b><br>{num}")
        else:
            tmp_text.append(num)



    z_text.append(tmp_text)
    z.append(tmp)


programming_experience = [item.split(".")[-1] for item in programming_experience]
ml_experience = [item.split(".")[-1] for item in ml_experience]
```

```
create_heatmap(z, ml_experience, programming_experience, z_text, "Oranges", "ML Experience in different responsibilities", subtitle="",
        xlabel="Experience in using Machine Learning", ylabel="Programming Experience")
```

8.63% 14.35% 18.51% 14.9% 16.78% 14.08% 12.76% 0 years < 1 years 1-3 years 3-5 years 5-10 years 10-20 years 20+ years00.050.10.1513.61% 21.46% 15.6% 11.61% 7.22% 7.59% 9.82% 4.44% 0 years < 1 years 1-2 years 2-3 years 3-4 years 4-5 years 5-10 years 10-20 years00.050.10.150.2

Professional subgroupsPython Is Essential for Data Analysis and Data Science.Years of Coding ExperienceYears of using ML Methods

0 years < 1 years 1-2 years 2-3 years 3-4 years 4-5 years 5-10 years 10-20 years 20+ years 20+ years 10-20 years 5-10 years 3-5 years 1-3 years < 1 years

ML Experience in different responsibilitiesExperience in using Machine LearningProgramming ExperienceInTransition86InTransition11912611790136ML Experts224ML Experts2620InTransition126InTransition162167147115130ML Experts320ML Experts1130149Mid Level184Mid Level188Mid Level223180270314180144Mid Level241Mid Level278Mid Level2912311362770Begginers247Begginers5265842663814710Begginers486Begginers720761234130

In the figure above we can also see a categorization of the professionals:

- The first group is the **Beginners - Juniors**. They have less than 3 years of experience in both coding and ML methods and they make up around **21.8%** of all the professionals who participated in the survey.
- The second group are **Coders in transition** (**5.4%**). Those people have decades-long coding experience for working with data, however, they have started working with machine learning only recently. These may be for example software engineers transitioning into data engineers or Machine Learning Engineers.
- The third category in the lower right corner is the **Machine Learning Experts** (**~10%**). Those people have been coding since long before the current AI revolution - with 10 or even over 20 years of both ML and coding experience, they may have started to specialize in the topic around the 2000s or even late 1990s. These people were doing machine learning before it was hype.
- The last group is the **Mid Level Data Scientists or ML Engineers** (**~15.4%**) with a solid understanding of ML concepts and a strong coding background.

So, to help you get your dream job in the AI and Data Science field, especially if you belong to the Beginners or Coders in Transition group I analyze below the top skills required for working with data and Machine Learning.

unfold_lessHide code

In [42]:

```
languages_columns = [col for col in scope_df.columns if col.startswith("Q12")]

languages_columns = languages_columns[0:len(languages_columns)-2]

x = list(scope_df[scope_df["Q23"] != "Other"]["Q23"].apply(lambda x : x.split("(")[0]).unique())

y = []

for col in languages_columns:

    y.append(scope_df[col].value_counts().index[0])



z = []




for col in languages_columns:

    tmp = []

    for role in list(scope_df[scope_df["Q23"] != "Other"]["Q23"].unique()) :

        if len(scope_df[scope_df["Q23"] == role][col].value_counts().values) > 0:

            languages_usage = scope_df[scope_df["Q23"] == role][col].value_counts().values[0]

        else:

            languages_usage = 0.00

        tmp.append(round(( languages_usage / scope_df[scope_df["Q23"] == role].shape[0]),2))
```

```
    z.append(tmp)


fig = go.Figure(data=go.Heatmap(
                   z=z,
                   x=x,
                   y=y,
    colorscale='YlorBr',
                   ))


large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Essential Programming Languages per Role</span>"


layout = dict(
    title = large_title_format,
    font = dict(color = '#7b6b59'),

)




fig['layout'].update(layout)
fig.update_traces(text=z, texttemplate="%{text}")


fig.show()
```

0.940.80.840.780.740.90.980.670.760.620.730.570.860.280.080.290.140.240.160.10.110.270.530.230.210.160.590.540.220.420.590.740.40.270.30.370.460.560.730.070.180.180.190.050.110.10.080.290.070.110.070.130.040.210.070.150.030.10.060.020.080.060.080.130.240.080.250.220.220.070.150.20.110.280.110.120.060.140.090.310.10.360.060.180.140.070.210.040.130.10.320.10.40.10.360.090.180.170.070.160.060.160.160.280.130.150.160.150.040.230.260.050.050.040.10.070.190.030.110.050.190.040.060.040.020.10.030.070.090.110.070.050.230.050.050.060.090.140.230.10.050.010.050.020.010.050.030.010.010.020.010.020.020.0100.050.010.060.010.030.010.030.0300.010.010.0200.08Data ScientistSoftware EngineerResearch ScientistDeveloper AdvocateData Analyst Data EngineerMachine Learning/ MLops EngineerEngineer Teacher / professorStatisticianManager Data AdministratorData ArchitectPythonRSQLCC#C++JavaJavascriptBashPHPMATLABJuliaGo

00.20.40.60.8Essential Programming Languages per Role

Regarding the most important programming language that you need to know, it's pretty obvious that is Python. You can see in the table above that **Python** is required for each role, along with **SQL** most of the time. Statisticians should also have R knowledge while Software Engineers and Developers might also work with Java and Javascript.

If you are thinking to become a Machine Learning Engineer, a Data Architect, or a Data Scientist then it would be beneficial to get familiarized with Cloud technologies since these roles require working with cloud computing platforms and other cloud services.

unfold_lessHide code

In [43]:

```
cloud_usage = scope_df.groupby(["Q23", "Cloud_usage"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts"})



top_labels = ['Yes', 'No']

colors = ['#a43725', '#cfbd9b']



x_data = []
```

```
for role in list(scope_df["Q23"].unique()):
    yes = cloud_usage[
        (cloud_usage["Q23"] == role) &
        (cloud_usage["Cloud_usage"] == "Yes")
    ].iloc[0]['counts']
    no = cloud_usage[
        (cloud_usage["Q23"] == role) &
        (cloud_usage["Cloud_usage"] == "No")
    ].iloc[0]['counts']
    sum_total = yes + no
    x_data.append([round( (yes /sum_total) * 100, 2), round( (no /sum_total) * 100, 2) ])


y_data = list(scope_df["Q23"].apply(lambda x : x.split("(")[0]).unique())


fig = go.Figure()


for i in range(0, len(x_data[0])):
    for xd, yd in zip(x_data, y_data):
        fig.add_trace(go.Bar(
            x=[xd[i]], y=[yd],
            orientation='h',
            marker=dict(
                color=colors[i],
                line=dict(color='rgb(248, 248, 249)', width=1)
```

```
    )
  ))



large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Cloud Usage by Role</span>"
small_title_format = "<span style='font-size:14px; font-family:Helvetica'></b></span>"



fig.update_layout(
  xaxis=dict(
    showgrid=False,
    showline=False,
    showticklabels=False,
    zeroline=False,
    domain=[0.15, 1]
  ),
  yaxis=dict(
    showgrid=False,
    showline=False,
    showticklabels=False,
    zeroline=False,
  ),
  title = large_title_format + "<br>" + small_title_format,
```

```
    font = dict(color = '#7b6b59'),
    barmode='stack',
    paper_bgcolor='white',
    plot_bgcolor='white',
    margin=dict(l=120, r=10, t=140, b=80),
    showlegend=False,
)

annotations = []

for yd, xd in zip(y_data, x_data):
    # labeling the y-axis
    annotations.append(dict(xref='paper', yref='y',
                            x=0.14, y=yd,
                            xanchor='right',
                            text=str(yd),
                            font=dict(family='Arial', size=14,
                                      color='rgb(67, 67, 67)'),
                            showarrow=False, align='right'))
    # labeling the first percentage of each bar (x_axis)
    annotations.append(dict(xref='x', yref='y',
                            x=xd[0] / 2, y=yd,
                            text=str(xd[0]) + '%',
                            font=dict(family='Arial', size=14,
```

```
                        color='rgb(248, 248, 255)'),
                    showarrow=False))
# labeling the first Likert scale (on the top)
if yd == y_data[-1]:
    annotations.append(dict(xref='x', yref='paper',
                        x=xd[0] / 2, y=1.1,
                        text=top_labels[0],
                        font=dict(family='Arial', size=14,
                            color='rgb(67, 67, 67)'),
                        showarrow=False))
space = xd[0]
for i in range(1, len(xd)):
        # labeling the rest of percentages for each bar (x_axis)
        annotations.append(dict(xref='x', yref='y',
                            x=space + (xd[i]/2), y=yd,
                            text=str(xd[i]) + '%',
                            font=dict(family='Arial', size=14,
                                color='rgb(248, 248, 255)'),
                            showarrow=False))
        # labeling the Likert scale
        if yd == y_data[-1]:
            annotations.append(dict(xref='x', yref='paper',
                                x=space + (xd[i]/2), y=1.1,
                                text=top_labels[i],
```

```
                        font=dict(family='Arial', size=14,
                                  color='rgb(67, 67, 67)'),
                        showarrow=False))
    space += xd[i]

fig.update_layout(annotations=annotations)

fig.show()
```

Cloud Usage by RoleData Scientist57.5%42.5%Software Engineer42.84%57.16%Research Scientist43.39%56.61%Other24.6%75.4%Developer Advocate44.07%55.93%Data Analyst 37.87%62.13%Data Engineer56.3%43.7%Machine Learning/ MLops Engineer67.79%32.21%Engineer 25.49%74.51%Teacher / professor36.59%63.41%Statistician28.46%71.54%Manager 52.18%47.82%Data Administrator37.14%62.86%Data Architect65.26%Yes34.74%No

Since machine learning and AI jobs entail the development of algorithms, let's have a look at the ML algorithms that an aspiring professional should know. The ones that are common for every role but especially for Data Scientists are **Linear or Logistic Regression** and **Decision Trees or Random Forests.** Data Scientists should also be able to use **Gradient Boosting Machines algorithms** while **Research Scientists** and **Machine Learning Engineers** should have a solid understanding of Deep Neural Networks since they use **Convolutional Neural Networks, MLPs, RNNs, and Transformers** on a regular basis.

unfold_lessHide code

In [44]:

```
roles_in_scope = [
    "Data Scientist",
    "Data Analyst (Business, Marketing, Financial, Quantitative, etc)",
    "Software Engineer",
```

```
    "Research Scientist",

    "Machine Learning/ MLops Engineer",

    "Data Engineer",

    "Statistician",

    "Data Architect"

]


ml_algorithms = [col for col in scope_df.columns if col.startswith("Q18")]

# Exclude None and others

ml_algorithms = ml_algorithms[:-2]

ml_algorithms_values = [scope_df[col].value_counts().index.to_list()[0].strip() for col in ml_algorithms]


x = [

    "Data Scientist",

    "Data Analyst",

    "Software Engineer",

    "Research Scientist",

    "Machine Learning/ MLops Engineer",

    "Data Engineer",

    "Statistician",

    "Data Architect"

]

y = ml_algorithms_values
```

```
z = []


for alogithm in ml_algorithms:

    tmp = []

    for role in roles_in_scope:

        tmp.append(round((scope_df[scope_df["Q23"] == role][alogithm].count() /
scope_df[scope_df["Q23"] == role].shape[0]),2))


    z.append(tmp)


fig = ff.create_annotated_heatmap(z, x=x, y=y, annotation_text=z,  colorscale='Oranges')


large_title_format = "<span style='font-size:30px; font-family:Times New Roman'> ML algorithms used
on a regular basis by job role</span>"


layout = dict(

    title = large_title_format,

    font = dict(color = '#7b6b59'),


)




fig['layout'].update(layout)

fig["layout"]["xaxis"].update(side="bottom")

fig.show()
```

Data ScientistData AnalystSoftware EngineerResearch ScientistMachine Learning/ MLops EngineerData EngineerStatisticianData ArchitectLinear or Logistic Regression Decision Trees or Random Forests Gradient Boosting Machines (xgboost, lightgbm, etc) Bayesian Approaches Evolutionary Approaches Dense Neural Networks (MLPs, etc) Convolutional Neural Networks Generative Adversarial Networks Recurrent Neural Networks Transformer Networks (BERT, gpt-3, etc) Autoencoder Networks (DAE, VAE, etc) Graph Neural Networks

ML algorithms used on a regular basis by job role0.760.490.50.590.640.610.620.610.710.390.390.470.550.50.460.540.590.210.240.320.480.320.240.310.270.110.150.270.220.180.220.210.060.020.040.090.060.030.040.030.250.070.160.30.410.130.080.120.350.130.320.470.630.290.110.380.060.020.060.110.140.040.030.080.240.080.160.240.330.150.110.190.220.050.110.20.360.090.050.130.10.010.050.170.180.040.040.050.080.040.050.130.10.080.030.05

When it comes to the Machine Learning Frameworks **Scikit-learn** is a must-have for Data Scientists and Machine Learning Engineers while **PyTorch**, **Tensorflow**, and **Keras** are used a lot by Machine Learning Engineers, Research Scientists, Data Architects, and Data Scientists for research and production needs.

unfold_lessHide code

In [45]:

```
roles_in_scope = [
    "Data Scientist",
    "Data Analyst (Business, Marketing, Financial, Quantitative, etc)",
    "Software Engineer",
    "Research Scientist",
    "Machine Learning/ MLops Engineer",
    "Data Engineer",
    "Statistician",
    "Data Architect"
```

```
]


ml_frameworks = [col for col in scope_df.columns if col.startswith("Q17")]

# Exclude None and others

ml_frameworks = ml_frameworks[:-2]

ml_frameworks_values = [scope_df[col].value_counts().index.to_list()[0].strip() for col in
ml_frameworks]


x = [

    "Data Scientist",

    "Data Analyst",

    "Software Engineer",

    "Research Scientist",

    "Machine Learning/ MLops Engineer",

    "Data Engineer",

    "Statistician",

    "Data Architect"

]

y = ml_frameworks_values


z = []


for framework in ml_frameworks:

    tmp = []

    for role in roles_in_scope:
```

```
        tmp.append(round((scope_df[scope_df["Q23"] == role][framework].count() /
scope_df[scope_df["Q23"] == role].shape[0]),2))

    z.append(tmp)

fig = ff.create_annotated_heatmap(z, x=x, y=y, annotation_text=z,  colorscale='Oranges')

large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>ML Frameworks used on a regular basis by job role</span>"

layout = dict(
    title = large_title_format,
    font = dict(color = '#7b6b59'),

)

fig['layout'].update(layout)
fig["layout"]["xaxis"].update(side="bottom")

fig.show()
```

Data ScientistData AnalystSoftware EngineerResearch ScientistMachine Learning/ MLops EngineerData EngineerStatisticianData ArchitectScikit-learn TensorFlow Keras PyTorch Fast.ai Xgboost LightGBM CatBoost Caret Tidymodels JAX PyTorch Lightning Huggingface

ML Frameworks used on a regular basis by job role0.820.460.530.620.780.610.380.530.470.240.410.450.620.40.20.450.430.190.330.380.560.30.160.350.340.140.270.410.570.270.130.380.050.010.040.040.10.040.020.040.50.180.190.240.430.280.190.280.270.070.080.120.220.140.110.070.150.030.040.060.140.080.070.050.090.040.020.070.040.030.130.040.060.030.010.050.010.010.070.050.010.00.010.030.040.010.020.010.070.020.050.080.150.040.020.120.170.020.070.110.270.070.030.05

Data science team sizing

Here I look at the relationship between company and Data Science team size. It seems that larger companies have bigger data science teams.

unfold_lessHide code

In [46]:

```
company_size_df = scope_df.groupby(["Q25"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts"})

company_size_df["relative_percent"] = company_size_df.apply(lambda x : x["counts"] / scope_df.shape[0], axis = 1)

company_size_df = company_size_df.sort_values(by=["Q25"])

company_size_df["Q25"] = company_size_df["Q25"].apply(lambda x : x.split(".")[-1])



data_team_size_df = scope_df.groupby(["Q26"]).agg({"Q2" : "count"}).reset_index().rename(columns={"Q2": "counts"})

data_team_size_df["relative_percent"] = data_team_size_df.apply(lambda x : x["counts"] / scope_df.shape[0], axis = 1)

data_team_size_df = data_team_size_df.sort_values(by=["Q26"])

data_team_size_df["Q26"] = data_team_size_df["Q26"].apply(lambda x : x.split(".")[-1])
```

```
traces = dict()


# Creating the bar chart
trace_company_size = get_bar_plot_trace(
    company_size_df["Q25"].to_list(),
    company_size_df["relative_percent"].to_list(),
    np.round((company_size_df["relative_percent"] *100), decimals = 2),
    0,
    company_size_df.shape[0]-0,
    company_size_df["counts"].to_list(),
    orientation="v"
)


trace_team_size = get_bar_plot_trace(
    data_team_size_df["Q26"].apply(lambda x : x.split("(")[0]),
    data_team_size_df["relative_percent"].to_list(),
    np.round((data_team_size_df["relative_percent"] *100), decimals = 2),
    0,
    data_team_size_df.shape[0]-0,
    data_team_size_df["counts"].to_list(),
    orientation="v"
)
```

```
fig = make_subplots(

    rows=1,

    cols=2 ,

    shared_yaxes=False,

    shared_xaxes=True,

    horizontal_spacing = 0.20,

    vertical_spacing = 0.10,

    subplot_titles=("Company Size", "Data Science Team Size")

)

traces["company_size"] = trace_company_size

traces["team_size"] = trace_team_size


fig.append_trace(traces["company_size"],1,1)

fig.append_trace(traces["team_size"],1,2)


large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Company and DS Team Size</span>"


layout = dict(

    title = large_title_format + "<br>",

    font = dict(color = '#7b6b59'),

    showlegend = False,

    margin = dict(t=150,pad=6),

    plot_bgcolor='#fff',
```

```
    bargap = 0.10,
)

fig['layout'].update(layout)

fig.show()

data_teams = [item for item in map_data_team_size.values() if not pd.isnull(item)]
company_size = [item for item in map_company_size.values() if not pd.isnull(item)]

z = []

for team in data_teams:
    tmp = []

    for company in company_size:

        tmp.append((scope_df[(scope_df["Q25"] == company) & (scope_df["Q26"] == team)].shape[0]))

    z.append(tmp)


y = [item.split(".")[-1] for item in map_data_team_size.values() if not pd.isnull(item)]
```

```
test = ['0-49 employees',
 '50-249 employees',
 '250-999 employees',
 '1000-9,999 employees',
 '10,000 or more employees']



fig1 = ff.create_annotated_heatmap(z, x=test, y=y, colorscale='Oranges')
layout = go.Layout(
    xaxis= {"title": "Company Size (employees)"},
    yaxis= {"title": "Data Science Team Size"},
    font = dict(color = '#7b6b59'),

)

fig1.update_layout(layout)
fig1.show()
```

23.42% 17.2% 14.98% 20.76% 23.33% 0-49 employees 50-249 employees 250-999 employees 1000-9,999 employees 10,000 or more employees00.050.10.150.216.02% 19.96% 15.31% 12.55% 7.18% 2.88% 24.96% 0 1-2 3-4 5-9 10-14 15-19 20+00.050.10.150.20.25

Company and DS Team SizeCompany SizeData Science Team Size

0-49 employees50-249 employees250-999 employees1000-9,999 employees10,000 or more employees 0 1-2 3-4 5-9 10-14 15-19 20+

Company Size (employees)Data Science Team Size561263192216225850392208208157391344254255148189262251261178651251562011061438579063321292336461230

From the illustration above we can notice that there is a correlation between the company's size and the Data Science team's size. Smaller companies have mostly Data Science Teams of 1-2 individuals while the larger ones have a much bigger team of 20+ members meaning that each member will have concrete responsibilities and tasks.

What education do AI specialists need?

Education requirements for data science and machine learning professionals vary by position, employer, and industry. Some data science professionals hold a mix of education levels. For example, someone might earn a bachelor's in computer science and complete a data science bootcamp. Or, they might complete a bachelor's in an unrelated field and then earn a master's in data science.

Let's have a look at the highest level of education that the professionals of the Kaggle Survey have. Almost half of them (**43.51%**) hold a Master's degree while 24.76% have a Bachelor's degree. So, from my point of view, the Master's degree tends to be a must-have for the market.

unfold_lessHide code

In [47]:

```
education_df = scope_df.groupby(["Q8"]).agg({"Q2" : "count"}).reset_index().rename(columns={col: "Q8", "Q2": "counts"})



education_df["relative_percent"] = education_df.apply(lambda x : (x["counts"] / scope_df.shape[0]), axis = 1)

education_df = education_df.sort_values(by=["relative_percent"], ascending=True)



create_single_bar_plot(
```

```
    x_values=education_df["relative_percent"].to_list(),
    y_values=education_df["Q8"].to_list(),
    display_text=np.round((education_df["relative_percent"] *100), decimals = 2),
    top_n=2,
    rest_n=education_df.shape[0]-2,
    hovertext = education_df["counts"].to_list(),
    title="Educational Qualifications",
    subtitle="",
    orientation="h"
)
```

2.5% 3.4% 3.74% 5.85% 16.24% 24.76% 43.51% 0.0%10.0%20.0%30.0%40.0%No formal education past high schoolProfessional doctorateSome college/university study without earning a bachelor’s degreeI prefer not to answerDoctoral degreeBachelor’s degreeMaster’s degree

Educational Qualifications

unfold_lessHide code

In [48]:

```
education_roles = scope_df[
    (scope_df["Q8"] != "I prefer not to answer") &
    (scope_df["Q23"] != "Other")
]

education_roles['Education_level'] = education_roles.apply(lambda row:
categorize_education(row["Q8"]), axis=1)
```

```
education_roles = education_roles.groupby(["Education_level", "Q23"]).agg({"Q2" :
"count"}).reset_index().rename(columns={"Q2": "counts"})



role_choices = list(education_roles["Q23"].unique())

education_choices = [

    "Lower than Bachelor",

    "Bachelor",

    "Master",

    "Higher than Master"

]

x = []

for education_level in education_choices:

    x.extend([education_level] * len(role_choices))


marker_size = []

text_markers = []

for education in education_choices:

    for con in role_choices:

        try:

            per = (education_roles[

                (education_roles["Q23"] == con) &

                (education_roles["Education_level"] == education)

            ].iloc[0]["counts"] / education_roles[education_roles["Q23"] == con]["counts"].sum()) *100

            marker_size.append(per)
```

```
        text_markers.append(str(round(per, 1))+"%")

    except IndexError as e:

        marker_size.append(0)


roles = []

for role in role_choices:

    roles.append(role.split("(")[0])


trace = go.Scatter(

    x = x,

    y = roles*4,

    mode='markers+text',

    textposition="middle right",

    text=text_markers,

    name="",

marker=dict(color=["#325C6E"]*len(role_choices)+["#a43725"]*len(role_choices)+["#edc860"]*len(role_choices)+["#E6b6a4"]*len(role_choices), opacity=0.8, size = marker_size))



large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Should you pursue an associate degree in data science?</span>"

small_title_format = "<span style='font-size:14px; font-family:Helvetica'>Education Level count by group and role</b></span>"
```

```
layout = go.Layout(barmode='stack', margin=dict(l=200), height=1000, title = large_title_format + "<br>"
+ small_title_format,  font = dict(color = '#7b6b59'),

                    legend = dict(orientation="h", x=0.1, y=1.15), plot_bgcolor='#fff', paper_bgcolor='#fff',

                    showlegend=False)


fig = go.Figure(data=[trace], layout=layout)

iplot(fig)
```

12.7%7.9%6.7%10.8%4.1%20.8%10.6%5.0%7.3%2.3%7.7%4.4%3.8%38.1%35.7%33.7%35.1%21.9%35.8%33.6%26.5%21.6%8.1%41.2%15.9%3.8%44.4%49.0%48.3%47.5%53.2%30.2%45.1%54.1%56.9%26.8%42.3%54.0%28.5%4.8%7.5%11.2%6.6%20.8%13.2%10.8%14.4%14.3%62.8%8.8%25.7%63.8%Lower than BachelorBachelorMasterHigher than MasterData AdministratorData Analyst Data ArchitectData EngineerData ScientistDeveloper AdvocateEngineer Machine Learning/ MLops EngineerManager Research ScientistSoftware EngineerStatisticianTeacher / professor

Should you pursue an associate degree in data science?Education Level count by group and role

Data scientists typically need at least a bachelor's degree in computer science, data science, or a related field. However, many employers in this field prefer a master's degree in data science or a related discipline.

Data analysts and data engineers usually need a bachelor's degree. Becoming a data scientist or computer and information research scientist usually requires a master's.

unfold_lessHide code

In [49]:

```
education_countries = pd.merge(scope_df.rename(columns={"Q4": "country"}), countries_df,
on=["country"], how="left")
```

```
education_countries["continent"] = education_countries.apply(lambda x :
fix_map_country_continent(map_country_continent, x["country"], x["continent"]), axis = 1)

education_countries = education_countries[

    (education_countries["continent"].notnull())&

    (education_countries["Q8"] != "I prefer not to answer")

]


education_countries['Education_level'] = education_countries.apply(lambda row:
categorize_education(row["Q8"]), axis=1)


education_countries = education_countries.groupby(["Education_level", "continent"]).agg({"Q2" :
"count"}).reset_index().rename(columns={"Q2": "counts"})


continent_choices = list(education_countries["continent"].unique())

education_choices = [

    "Lower than Bachelor",

    "Bachelor",

    "Master",

    "Higher than Master"

]

x = []

for education_level in education_choices:

    x.extend([education_level] * len(continent_choices))
```

```
marker_size = []

text_markers = []

for education in education_choices:

    for con in continent_choices:

        try:

            per = (education_countries[

                (education_countries["continent"] == con) &

                (education_countries["Education_level"] == education)

            ].iloc[0]["counts"] / education_countries[education_countries["continent"] ==
con]["counts"].sum()) *100

            marker_size.append(per)

            text_markers.append(str(round(per, 1))+"%")

        except IndexError as e:

            marker_size.append(0)



trace = go.Scatter(

    x = x,

    y = continent_choices*4,

    mode='markers+text',

    textposition="middle right",

    text=text_markers,

    name="",

marker=dict(color=["#325C6E"]*len(continent_choices)+["#a43725"]*len(continent_choices)+["#edc860"
]*len(continent_choices)+["#E6b6a4"]*len(continent_choices), opacity=0.8, size = marker_size))
```

```
large_title_format = "<span style='font-size:30px; font-family:Times New Roman'>Education Level</span>"

small_title_format = "<span style='font-size:14px; font-family:Helvetica'>count by group and continent</b></span>"
```

```
layout = go.Layout(barmode='stack', margin=dict(l=200), height=600, title = large_title_format + "<br>" + small_title_format, font = dict(color = '#7b6b59'),

                    legend = dict(orientation="h", x=0.1, y=1.15), plot_bgcolor='#fff', paper_bgcolor='#fff',

                    showlegend=False)


fig = go.Figure(data=[trace], layout=layout)

iplot(fig)
```

6.6%8.2%5.2%7.4%6.1%35.1%23.9%31.8%12.7%28.8%43.2%44.9%45.8%50.9%40.9%15.1%23.0%17.3%29.1%24.2%Lower than BachelorBachelorMasterHigher than MasterAfricaAmericasAsiaEuropeOceania

Education Levelcount by group and continent

Artificial Intelligence salaries (by role, industry, education & more)

I hope the last part of the analysis to help you in your salary negotiations or when negotiating a job offer :P

So, the $100 Dollar Question: How Much Do Artificial Intelligence (AI) and Data Jobs Actually Pay?

Well, the exact numbers of AI salaries depend on many factors, including specific job responsibilities, industry, experience, education level, and geographic location.

Therefore, for the salary benchmarking I'll get each factor separately and do a salary comparison based on that. We would get more representative insights if I would take into account all of them at once, or jointly, for instance examine salaries based on industry and job roles, or based on country, industry, and job roles. However, I want to keep the analysis simple so let's do the deep dives by exploring each factor separately.

Starting with the analysis of the yearly compensation by job role, it is clear that the 1st best-paying salary is for **Data Architects (median at 65,000 US dollars per year)**, followed by **Managers (median at 55,000 US dollars per year)** and **Data Scientists**, earning slightly less **(median at 45,000 US dollars per year)** while **Statisticians** are paid less than any other profession.

Disclaimer: The exact numbers of the salaries might be not fully accurate because we have to take into consideration all the factors mentioned at the beginning of the section for the salary benchmarking instead of examining them one by one. But we can get an overview of the market trends in 2022.

unfold_lessHide code

In [50]:

```
scope_df[['min_w','max_w']]=scope_df['Q29'].str.replace('$', '', regex=False).str.replace(',', '',
regex=False).str.replace('>', '', regex=False).str.split('-', expand = True)

scope_df[['min_w','max_w']] = scope_df[['min_w','max_w']].astype('float')

scope_df['Mean_Compensation']=(scope_df['min_w']+scope_df['max_w'])/2 + 0.5


scope_df["Q23"] = scope_df["Q23"].apply(lambda x : x.split("(")[0]).to_list()


create_box_plot(scope_df,"Q23", "Mean_Compensation",  "Yearly compensation by profession")
```

Data ScientistSoftware EngineerResearch ScientistOtherDeveloper AdvocateData Analyst Data EngineerMachine Learning/ MLops EngineerEngineer Teacher / professorStatisticianManager Data AdministratorData Architect0100k200k300k400k500k600k700k

Yearly compensation by professionCompensation in USD

Moving on to the comparison by industry in the first place as it can be seen in the chart are the **Medical / Pharmaceutical** and **Insurance companies**, offering 45,000 US dollars yearly compensation on average.

Even if the numbers are not accurate, the trends though look reasonable. The Pharmaceutical and Health Sciences sector played a key role during the COVID-19 pandemic. To deal with the global crisis, traditional competitors teamed up to accelerate research, and this “new normal” mindset triggered organizations to rethink their operational models.

unfold_lessHide code

In [51]:

```
create_box_plot(scope_df,"Q24", "Mean_Compensation",  "Yearly compensation by industry")
```

Online Service/Internet-based ServicesInsurance/Risk AssessmentGovernment/Public ServiceManufacturing/FabricationComputers/TechnologyAccounting/FinanceAcademics/EducationNon-profit/ServiceOtherMedical/PharmaceuticalMarketing/CRMEnergy/MiningBroadcasting/Communications Retail/SalesShipping/Transportation0100k200k300k400k500k600k700k

Yearly compensation by industryCompensation in USD

As you might expect there's a clear correlation between education level and salary. Generally, it seems that the more educated you are, the greater your salary becomes.

The same applies to years of coding experience or ML experience.

unfold_lessHide code

In [52]:

```
scope_df['Education_level'] = scope_df.apply(lambda row: categorize_education(row["Q8"]), axis=1)


map_education = {
    "Lower than Bachelor": "1. Lower than Bachelor" ,
    "Bachelor": "2. Bachelor",
    "Master": "3. Master",
    "Higher than Master": "4. Higher than Master",
    "Other": "Other"

}

results = scope_df
results["Education_level"] = results["Education_level"].apply(lambda x : map_education[x])
results = results.sort_values(by=["Education_level"])
results["Education_level"] = results["Education_level"].apply(lambda x : x.split(".")[-1].strip()).to_list()
results = results[results["Education_level"] != "Other"]


create_box_plot(results,"Education_level", "Mean_Compensation",  "Yearly compensation by education level")
```

Lower than BachelorBachelorMasterHigher than Master0100k200k300k400k500k600k700k

Yearly compensation by education levelCompensation in USD

unfold_lessHide code

In [53]:

```
tmp = scope_df.sort_values(by=["Q11"])

tmp = tmp[tmp["Q11"].notnull()]

tmp["Q11"] = tmp["Q11"].apply(lambda x : x.split(".")[-1])



create_box_plot(tmp, "Q11", "Mean_Compensation",  "Yearly compensation by years of coding experience")



tmp = scope_df.sort_values(by=["Q16"])

tmp = tmp[tmp["Q16"].notnull()]

tmp["Q16"] = tmp["Q16"].apply(lambda x : x.split(".")[-1])



create_box_plot(tmp, "Q16", "Mean_Compensation",  "Yearly compensation by years of ML experience")
```

0 years < 1 years 1-3 years 3-5 years 5-10 years 10-20 years 20+ years0100k200k300k400k500k600k700k

Yearly compensation by years of coding experienceCompensation in USD

0 years < 1 years 1-2 years 2-3 years 3-4 years 4-5 years 5-10 years 10-20 years0100k200k300k400k500k600k700k

Yearly compensation by years of ML experienceCompensation in USD

In terms of continent, it seems that the **Americas and Oceania pay higher salaries for AI jobs compared to Europe, Asia, and Africa.**

unfold_lessHide code

In [54]:

```
education_countries = pd.merge(scope_df.rename(columns={"Q4": "country"}), countries_df, on=["country"], how="left")

education_countries["continent"] = education_countries.apply(lambda x : fix_map_country_continent(map_country_continent, x["country"], x["continent"]), axis = 1)

create_box_plot(education_countries, "continent", "Mean_Compensation",  "Yearly compensation by continent")
```

EuropeOceaniaAsiaAmericasAfrica0100k200k300k400k500k600k700k

Yearly compensation by continentCompensation in USD

Another clear trend is that large companies pay higher wages. One explanation could be that workers in big firms are more skilled.

unfold_lessHide code

In [55]:

```
tmp = scope_df.sort_values(by=["Q25"])

tmp = tmp[tmp["Q25"].notnull()]

tmp["Q25"] = tmp["Q25"].apply(lambda x : x.split(".")[-1])

create_box_plot(tmp, "Q25", "Mean_Compensation",  "Yearly compensation by company size")
```

0-49 employees 50-249 employees 250-999 employees 1000-9,999 employees 10,000 or more employees0100k200k300k400k500k600k700k

Yearly compensation by company sizeCompensation in USD

Conclusion

All in all, my goal through this analysis was to provide insights about the state of AI adoption & MLOps in Industry, by examing to what extent enterprises have Machine Learning models in production, what are the main tools that they use for Data Storage, Model training, deployment, and other processes, what are the main frameworks and libraries used on a regular basis as well as what are the most common AI job roles that the companies seek.

Key Takeaways

- **21.7%** of the professionals in the survey said that their companies **haven't started yet to explore Machine Learning methods** vs **32.8%** of the respondents who stated that their organizations have already **Machine Learning models in production** either in advanced or in an intermediate stage.
- **Online / Internet-based Services**, **insurances**, and **tech** companies are the leaders in the adoption of Artificial Intelligence.
- Even if smaller companies might be better candidates for the implementation of AI, due to the absence of legacy systems, the survey results show that big companies are leading at the moment the way in AI adoption.
- **45%** of the professional that participated in the survey use **Cloud Computing Platforms** with Amazon Web Services (AWS) and Google Cloud Platform (GCP) being the dominant ones in 2022.
- The most popular AI jobs are Data Scientist and Data Analyst.
- **Top Skills Required for a Data Scientist / Machine Learning Engineer:**
    - **Programming Languages:** Python, SQL
    - **Machine Learning Frameworks:** Scikit-learn, Tensorflow, Keras
    - **Machine Learning Algorithms:** Linear and Logistic Regression, Decision Trees, Gradient Boosting Machines, CNNs, MLPs, Transformers
    - **Experience using Cloud Computing Platforms**
    - **Data Visualization Libraries:** Matplotlib, Seaborn, Plotly
- The main responsibilities of a **Data Scientist** are:
    - Analyze and understand data to influence product or business decisions
    - Build prototypes to explore applying machine learning to new areas
    - Experimentation and iteration to improve existing ML models

      while for a **Machine Learning Engineer:**
        - Build prototypes to explore applying machine learning to new areas
        - Experimentation and iteration to improve existing ML models

    - Build and/or run a machine learning service that operationally improves the products or workflows
- **43.51%** of the professionals hold a Master's degree
- Transfer Learning methods used mainly in Computer Vision Tasks
- Only **31.3%** of the respondents **use specialized hardware when training machine learning models** which indicates either that usually we don't deal with big data or deep neural networks that require huge resources for training or that the companies don't invest in specialized hardware and this causes a bottleneck to the productionization of ML models.

References

1. Mckinsey Report: The state of AI in 2021
2. Global Cloud Computing Market Report 2022: Increased Resource, User Mobility, and Ongoing Migration of Applications Over the Cloud Driving Growth - ResearchAndMarkets.com
3. ML Operationalization: Building a path to real-world business success
4. Kaggle Notebook: Spending dollars for MS in Data Science - Worth it ?
5. Kaggle Notebook: A story told through a heatmap
6. Kaggle Notebook: Data Science in 2021 : Adaptation or Adoption?
7. Kaggle Notebook: Head in the Clouds
8. What is Cloud Computing? The Key to Putting Models into Production