Our goal in this project is to use Data Science Stack Exchange to determine what content should a data science education company create, based on interest by subject.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
questions = pd.read_csv("questions.csv", parse_dates=["CreationDate"])
```

In [3]:
```
questions.info()
```

Out[3]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8839 entries, 0 to 8838
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              8839 non-null   int64
 1   CreationDate    8839 non-null   datetime64[ns]
 2   Score           8839 non-null   int64
 3   ViewCount       8839 non-null   int64
 4   Tags            8839 non-null   object
 5   AnswerCount     8839 non-null   int64
 6   FavoriteCount   2831 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(4), object(1)
memory usage: 483.5+ KB
```

In [4]:
```
questions.fillna(value="FavoriteCount", 0), inplace=True)
questions["FavoriteCount"] = questions["FavoriteCount"].astype(int)
```

In [5]:
```
questions["Tags"] = questions["Tags"].str.replace("<|>", "").str.split("><")
```

Out[5]:
```
C:\Users\ok\AppData\Local\Temp\ipykernel_4992\833902828.py:1: FutureWarning: The default value of regex will ch
ange from True to False in a future version.
  questions["Tags"] = questions["Tags"].str.replace("<|>", "").str.split("><")
```

Most Used and Most Viewed

In [6]:
```
tag_count = dict()

for tags in questions["Tags"]:
    for tag in tags:
        if tag in tag_count:
            tag_count[tag] += 1
        else:
            tag_count[tag] = 1
```

In [7]:
```
tag_count = pd.DataFrame.from_dict(tag_count, orient="index")
tag_count.rename(columns={0: "Count"}, inplace=True)
tag_count.head(10)
```

Out[7]:

	Count
machine-learning	2693
data-mining	217
regression	347
linear-regression	175
regularization	50
python	1814
time-series	466
forecast	34
forecasting	85
scikit-learn	540

In [8]:
```
most_used = tag_count.sort_values(by="Count").tail(20)
most_used
```

Out[8]:

	Count
machine-learning-model	224
statistics	234
clustering	257
predictive-modeling	265
r	268
dataset	340
regression	347
pandas	354
lstm	402
time-series	466
cnn	489
nlp	493
scikit-learn	540
tensorflow	584
classification	685
keras	935
neural-network	1055
deep-learning	1220
python	1814
machine-learning	2693

In [9]:
```
tag_view_count = dict()

for index, row in questions.iterrows():
    for tag in row["Tags"]:
        if tag in tag_view_count:
            tag_view_count[tag] += row["ViewCount"]
        else:
            tag_view_count[tag] = row["ViewCount"]

tag_view_count = pd.DataFrame.from_dict(tag_view_count, orient="index")
tag_view_count.rename(columns={0: "ViewCount"}, inplace=True)

most_viewed = tag_view_count.sort_values(by="ViewCount").tail(20)
```

In [10]:
```
fig, axes = plt.subplots(nrows=1, ncols=2)
fig.set_size_inches(24, 10)
most_used.plot(kind="barh", sax=axes[0], subplots=True)
most_viewed.plot(kind="barh", ax=axes[1], subplots=True)
plt.show()
```


In [20]:
```
in_used = pd.merge(most_used, most_viewed, how="left", left_index=True, right_index=True)
in_viewed = pd.merge(most_used, most_viewed, how="right", left_index=True, right_index=True)
```

Relations Between Tags

In [11]:
```
all_tags = list(tag_count.index)
```

In [22]:
```
associations = pd.DataFrame(index=all_tags, columns=all_tags)
associations.iloc[0:4,0:4]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_4992\3025529393.py in <module>
----> 1 associations = pd.DataFrame(index=all_tags, columns=all_tags)
      2 associations.iloc[0:4,0:4]

NameError: name 'all_tags' is not defined
```

In [23]:
```
associations.fillna(0, inplace=True)

for tags in questions["Tags"]:
    associations.loc[tags, tags] += 1
```

This dataframe is quite large. We will just focus our attention on the most used tags.

In [25]:
```
relations_most_used = associations.loc[most_used.index, most_used.index]
```

In [26]:
```
for i in range(relations_most_used.shape[0]):
    relations_most_used.iloc[i,i] = pd.np.NaN
```

```
C:\Users\Waqas\AppData\Local\Temp\ipykernel_19808\1217242379.py:1: FutureWarning: The pandas.np module is depre
cated and will be removed from pandas in a future version. Import numpy directly instead.
  relations_most_used.iloc[i,i] = pd.np.NaN
```

In [27]:
```
plt.figure(figsize=(12,8))
sns.heatmap(relations_most_used, cmap="Oceans", annot=False)
plt.show()
```


At the glance of an eye, someone with sufficient domain knowledge can tell that the most popular topic at the moment, as shown by our analysis, is **deep learning**. Lets prove this with **all_questions** dataset

In [28]:
```
all_q = pd.read_csv("all_questions.csv", parse_dates=["CreationDate"])
```

In [29]:
```
all_q["Tags"] = all_q["Tags"].str.replace("<|>", "").str.split("><")
```

```
C:\Users\Waqas\AppData\Local\Temp\ipykernel_19808\2212242379.py:1: FutureWarning: The default value of regex wi
ll change from True to False in a future version.
  all_q["Tags"] = all_q["Tags"].str.replace("<|>", "").str.split("><")
```

Deep learning includes `["lstm", "cnn", "scikit-learn", "tensorflow", "keras", "neural-network", "deep-learning"]` .

We'll now create a function that assigns `1` to deep learning questions and `0` otherwise, and we use it.

In [30]:
```
def class_deep_learning(tags):
    for tag in tags:
        if tag in ["lstm", "cnn", "scikit-learn", "tensorflow","keras", "neural-network", "deep-learning"]:
            return 1
    return 0
```

In [31]:
```
all_q["DeepLearning"] = all_q["Tags"].apply(class_deep_learning)
```

In [32]:
```
all_q.head()
```

Out[32]:

	Id	CreationDate	Tags	DeepLearning
0	45416	2019-02-12 00:36:29	[python, keras, tensorflow, cnn, probability]	1
1	45418	2019-02-12 00:50:39	[neural-network]	1
2	45422	2019-02-12 04:40:51	[python, ibm-watson, chatbot]	0
3	45426	2019-02-12 04:51:49	[keras]	1
4	45427	2019-02-12 05:08:24	[r, predictive-modeling, machine-learning-mode...	0

Since we don't have all the data for the first quarter of 2020, we'll get rid of those dates:

In [33]:
```
all_q = all_q[all_q["CreationDate"].dt.year < 2020]
```

Let's create a column that identifies the quarter in which a question was asked.

In [34]:
```
def fetch_quarter(datetime):
    year = str(datetime.year)[-2:]
    quarter = str((datetime.month-1) // 3 + 1)
    return "{}Q{}".format(year, quarter)

all_q["Quarter"] = all_q["CreationDate"].apply(fetch_quarter)
```

In [35]:
```
all_q.head()
```

Out[35]:

	Id	CreationDate	Tags	DeepLearning	Quarter
0	45416	2019-02-12 00:36:29	[python, keras, tensorflow, cnn, probability]	1	19Q1
1	45418	2019-02-12 00:50:39	[neural-network]	1	19Q1
2	45422	2019-02-12 04:40:51	[python, ibm-watson, chatbot]	0	19Q1
3	45426	2019-02-12 04:51:49	[keras]	1	19Q1
4	45427	2019-02-12 05:08:24	[r, predictive-modeling, machine-learning-mode...	0	19Q1

In [36]:
```
quarterly = all_q.groupby('Quarter').agg({'DeepLearning': ['sum', 'size']})

quarterly.columns = ['DeepLearningQuestions', 'TotalQuestions']

quarterly['DeepLearningRate'] = quarterly['DeepLearningQuestions']/quarterly['TotalQuestions']

quarterly.reset_index(inplace=True)
quarterly.head()
```

Out[36]:

	Quarter	DeepLearningQuestions	TotalQuestions	DeepLearningRate
0	14Q2	9	157	0.057325
1	14Q3	13	189	0.068783
2	14Q4	21	216	0.097222
3	15Q1	18	190	0.094737
4	15Q2	28	284	0.098592

In [38]:
```
ax1 = quarterly.plot(x="Quarter", y="DeepLearningRate",kind="line", linestyle="-", marker="o", color="orange",
                     figsize=(24,12))

ax2 = quarterly.plot(x="Quarter", y="TotalQuestions",kind="bar", ax=ax1, secondary_y=True, alpha=0.7, rot=45)

for idx, t in quarterly["TotalQuestions"].iteritems():
    ax2.text(idx, t, str(t), ha="center", va="bottom") # alias of ax1.get_xlim()

ax1.get_legend().remove()

handles1, labels1 = ax1.get_legend_handles_labels()
handles2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(handles=handles1 + handles2,labels=labels1 + labels2,loc="upper left", prop={"size": 12})

for ax in (ax1, ax2):
    for l in ["top", "right"]:
        ax.spines[l].set_visible(False)
    ax1.tick_params(right=False, labelright=False)
```


It seems that deep learning questions was a high-growth trend since the start of DSSE and it looks like it is plateauing. There is no evidence to suggest that interest in deep learning is decreasing and so we maintain our previous idea of proposing that we create deep learning content.