Amazon Lex Bot - Stock Advice
Group 4

Carl Buchholz
Angela Maturo
Alex Eakins
Jean-Pierre Koudifo

Summary

- Use an Amazon Lex Bot to provide stock advice based on a person’s personality type

- Organize and clean data to determine 4 different personality types

- Use machine learning to build a LSTM RNN model to predict future stock prices of 10 tech stocks

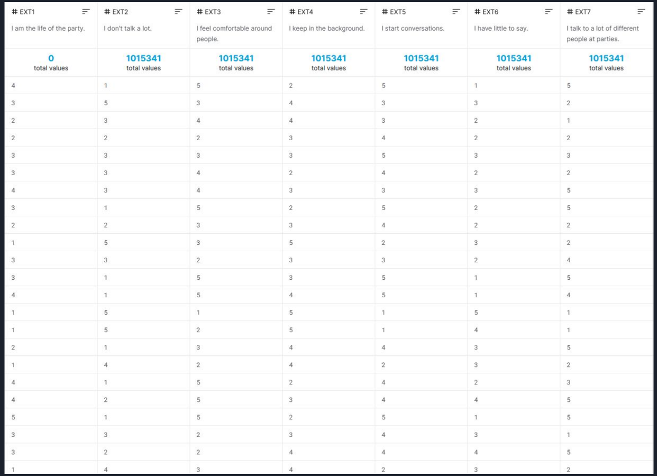
Risk Tolerance and Personality Type

Data from Kaggle - Big Five Personality Test

The Big Five personality traits, also known as the five-factor model (FFM) and the OCEAN model, is a taxonomy, or grouping, for personality traits. When factor analysis (a statistical technique) is applied to personality survey data, some words used to describe aspects of personality are often applied to the same person.

About the Personality Data

- Collected (2016-2018) through an interactive online personality test.
- The personality test was constructed from the IPIP.
- This dataset contains 1,015,342 questionnaire answers collected online by Open Psychometrics.
- The scale was labeled between 1=Disagree, 3=Neutral, 5=Agree



# EXT1	# EXT2	# EXT3	# EXT4	# EXT5	# EXT6	# EXT7
I am the life of the party.	I don't talk a lot.	I feel comfortable around people.	I keep in the background.	I start conversations.	I have little to say.	I talk to a lot of different people at parties.
0 total values	1015341 total values	1015341 total values	1015341 total values	1015341 total values	1015341 total values	1015341 total values
4	1	5	2	5	1	5
3	5	3	4	3	3	2
2	3	4	4	3	2	1
2	2	2	3	4	2	2
3	3	3	3	5	3	3
3	3	4	2	4	2	2
4	3	4	3	3	3	5
3	1	5	2	5	2	5
2	2	3	3	4	2	2
1	5	3	5	2	3	2
3	3	2	3	3	2	4
3	1	5	3	5	1	5
4	1	5	4	5	1	4
1	5	1	5	1	5	1
1	5	2	5	1	4	1
2	1	3	4	4	3	5
1	4	2	4	2	3	2
4	1	5	2	4	2	3
4	2	5	3	4	4	5
5	1	5	2	5	1	5
3	3	2	3	4	3	1
3	2	2	4	4	4	5
1	4	3	4	2	3	2

Personality Types

Conscientious   [Risk Tolerance: Low]

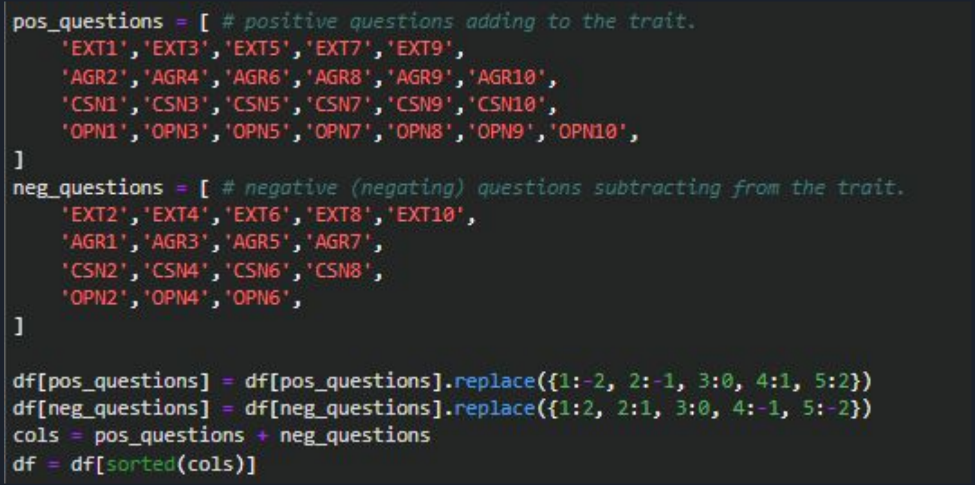
Agreeableness [Risk Tolerance: Moderate]

Open Personality [Risk Tolerance: Moderate to High]

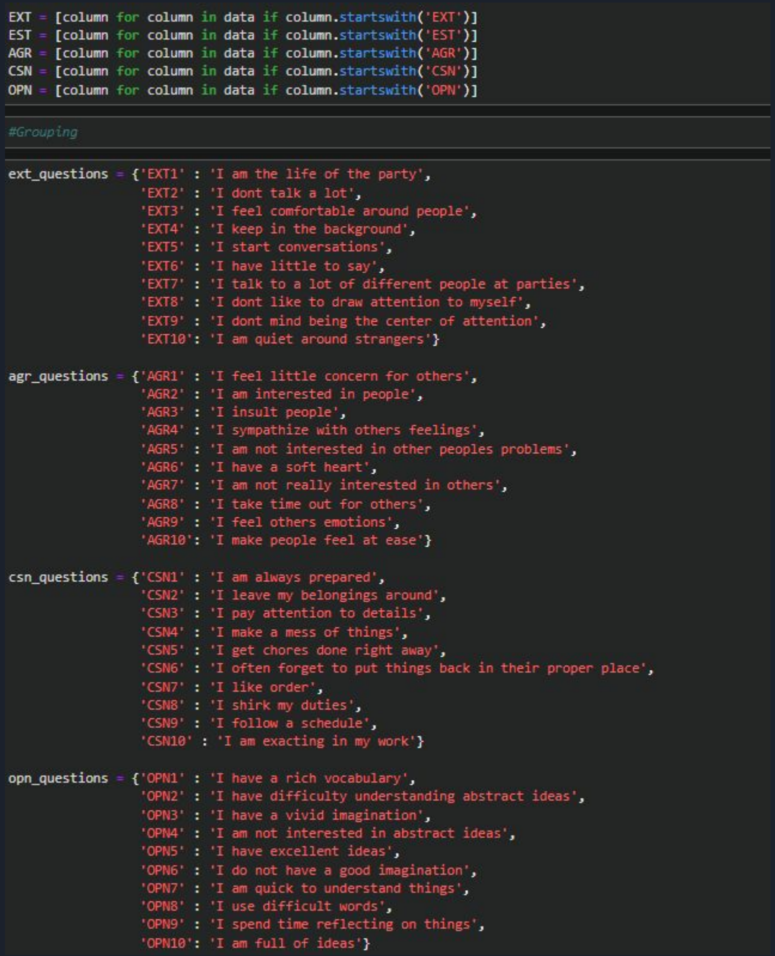
Extraversion [Risk Tolerance: High]

Organizing/Cleaning Data

- Grouped Questions
- Restructured Data



```
pos_questions = [ # positive questions adding to the trait.
    'EXT1','EXT3','EXT5','EXT7','EXT9',
    'AGR2','AGR4','AGR6','AGR8','AGR9','AGR10',
    'CSN1','CSN3','CSN5','CSN7','CSN9','CSN10',
    'OPN1','OPN3','OPN5','OPN7','OPN8','OPN9','OPN10',
]
neg_questions = [ # negative (negating) questions subtracting from the trait.
    'EXT2','EXT4','EXT6','EXT8','EXT10',
    'AGR1','AGR3','AGR5','AGR7',
    'CSN2','CSN4','CSN6','CSN8',
    'OPN2','OPN4','OPN6',
]

df[pos_questions] = df[pos_questions].replace({1:-2, 2:-1, 3:0, 4:1, 5:2})
df[neg_questions] = df[neg_questions].replace({1:2, 2:1, 3:0, 4:-1, 5:-2})
cols = pos_questions + neg_questions
df = df[sorted(cols)]
```



```
EXT = [column for column in data if column.startswith('EXT')]
EST = [column for column in data if column.startswith('EST')]
AGR = [column for column in data if column.startswith('AGR')]
CSN = [column for column in data if column.startswith('CSN')]
OPN = [column for column in data if column.startswith('OPN')]
```

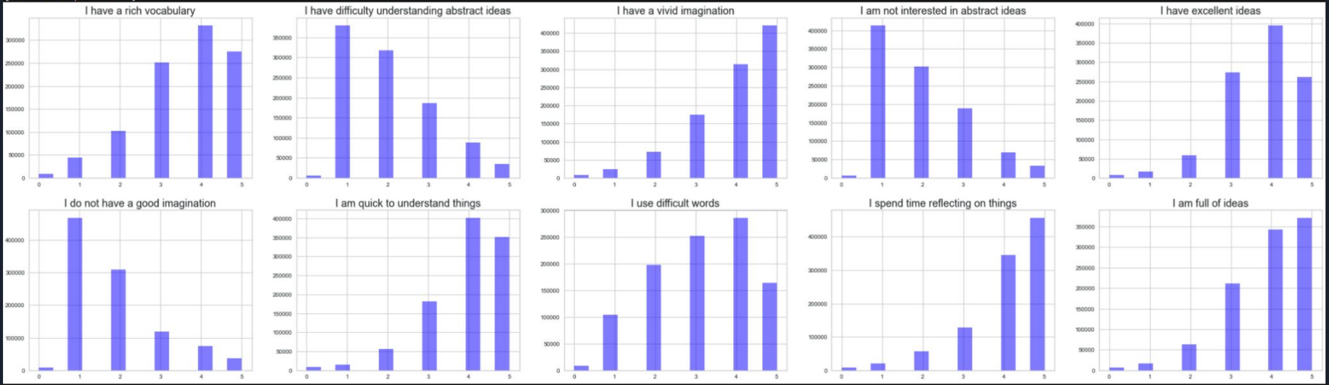
```
#Grouping
```

```
ext_questions = {'EXT1' : 'I am the life of the party',
                 'EXT2' : 'I dont talk a lot',
                 'EXT3' : 'I feel comfortable around people',
                 'EXT4' : 'I keep in the background',
                 'EXT5' : 'I start conversations',
                 'EXT6' : 'I have little to say',
                 'EXT7' : 'I talk to a lot of different people at parties',
                 'EXT8' : 'I dont like to draw attention to myself',
                 'EXT9' : 'I dont mind being the center of attention',
                 'EXT10': 'I am quiet around strangers'}

agr_questions = {'AGR1' : 'I feel little concern for others',
                 'AGR2' : 'I am interested in people',
                 'AGR3' : 'I insult people',
                 'AGR4' : 'I sympathize with others feelings',
                 'AGR5' : 'I am not interested in other peoples problems',
                 'AGR6' : 'I have a soft heart',
                 'AGR7' : 'I am not really interested in others',
                 'AGR8' : 'I take time out for others',
                 'AGR9' : 'I feel others emotions',
                 'AGR10': 'I make people feel at ease'}

csn_questions = {'CSN1' : 'I am always prepared',
                 'CSN2' : 'I leave my belongings around',
                 'CSN3' : 'I pay attention to details',
                 'CSN4' : 'I make a mess of things',
                 'CSN5' : 'I get chores done right away',
                 'CSN6' : 'I often forget to put things back in their proper place',
                 'CSN7' : 'I like order',
                 'CSN8' : 'I shirk my duties',
                 'CSN9' : 'I follow a schedule',
                 'CSN10' : 'I am exacting in my work'}

opn_questions = {'OPN1' : 'I have a rich vocabulary',
                 'OPN2' : 'I have difficulty understanding abstract ideas',
                 'OPN3' : 'I have a vivid imagination',
                 'OPN4' : 'I am not interested in abstract ideas',
                 'OPN5' : 'I have excellent ideas',
                 'OPN6' : 'I do not have a good imagination',
                 'OPN7' : 'I am quick to understand things',
                 'OPN8' : 'I use difficult words',
                 'OPN9' : 'I spend time reflecting on things',
                 'OPN10': 'I am full of ideas'}
```

Visualizing Correlation

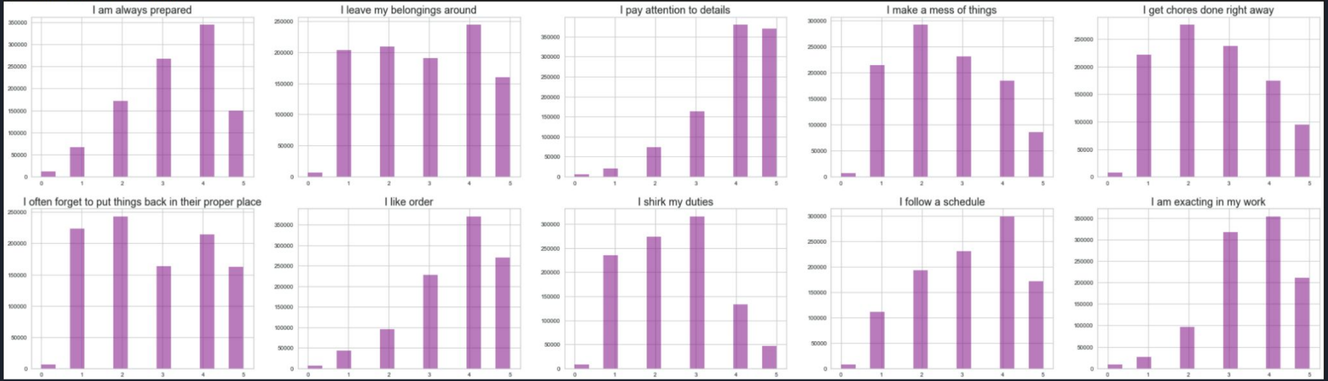
```
print('Q&As Related to Open Personality')
vis_questions(OPN, opn_questions, 'blue')
```

```
Q&As Related to Open Personality
```



```
print('Q&As Related to Conscientious Personality')
vis_questions(CSN, csn_questions, 'purple')
```

```
Q&As Related to Conscientious Personality
```

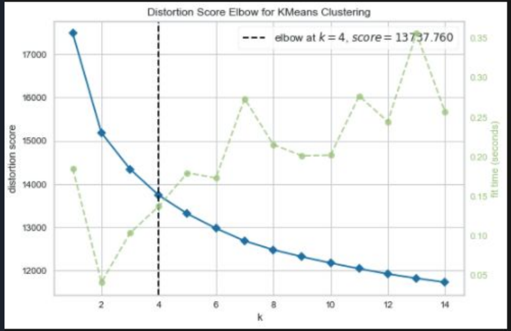


KMeans and Clustering

- Changed scale of data from 0-5 to 0-1
- Used Yellowbrick API to visualize K-Elbow to find the appropriate number of clusters
- Created cluster predictive model

```
from sklearn.preprocessing import MinMaxScaler

columns = list(df.columns)

scaler = MinMaxScaler(feature_range=(0,1))
df = scaler.fit_transform(df)
df = pd.DataFrame(df, columns=columns)
df_sample = df[:5000]
```

```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

kmeans = KMeans()
visualizer = KElbowVisualizer(kmeans, k=(1,15))
visualizer.fit(df_sample)
visualizer.poof()
```



```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel=
```

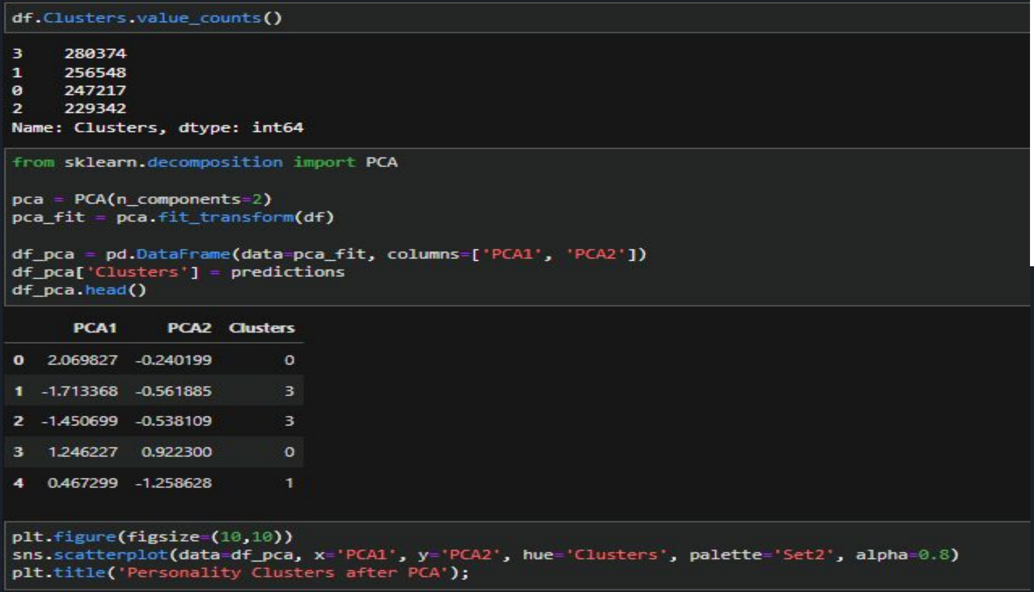
```
kmeans = KMeans(n_clusters=4)
k_fit = kmeans.fit(df)
```

```
pd.options.display.max_columns = 10
predictions = k_fit.labels_
df['Clusters'] = predictions
df.head()
```

	AGR1	AGR10	AGR2	AGR3	AGR4	...	Extroversion	Agreeableness	Conscientiousness	Openness	Clusters
0	0.75	0.75	1.00	0.75	0.75	...	0.900	0.725	0.550	0.875	0
1	1.00	0.50	0.75	1.00	1.00	...	0.250	0.850	0.675	0.625	3
2	1.00	0.50	0.75	1.00	0.75	...	0.375	0.800	0.600	0.775	3
3	0.75	0.75	0.75	0.50	0.75	...	0.400	0.700	0.375	0.725	0
4	1.00	0.50	1.00	1.00	1.00	...	0.475	0.900	0.950	0.950	1

PCA and Clustering

- Used Principal Component Analysis (PCA) to reduce the number of variables of a data set, while preserving as much information as possible.



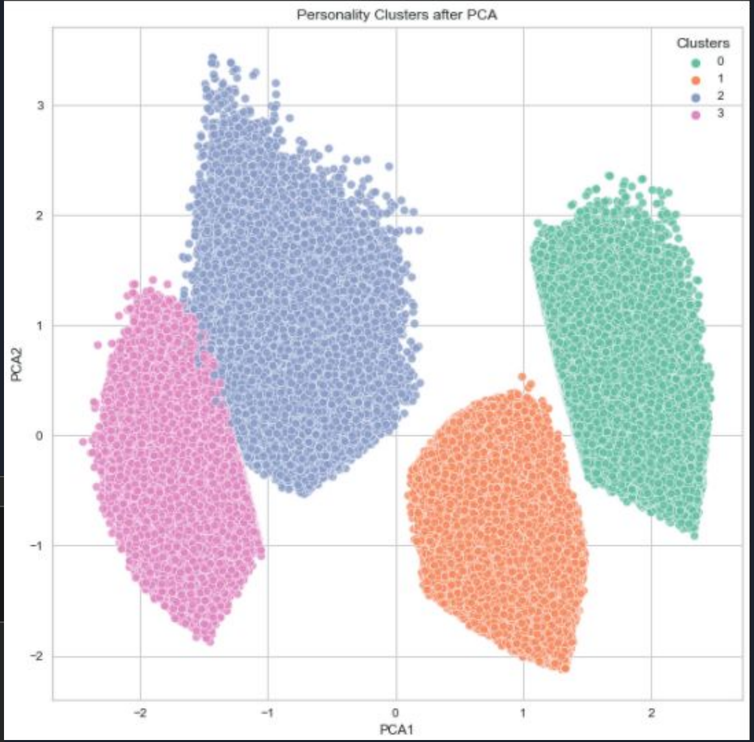
```
df.Clusters.value_counts()
```

```
3    280374
1    256548
0    247217
2    229342
Name: Clusters, dtype: int64
```

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_fit = pca.fit_transform(df)

df_pca = pd.DataFrame(data=pca_fit, columns=['PCA1', 'PCA2'])
df_pca['Clusters'] = predictions
df_pca.head()
```

	PCA1	PCA2	Clusters
0	2.069827	-0.240199	0
1	-1.713368	-0.561885	3
2	-1.450699	-0.538109	3
3	1.246227	0.922300	0
4	0.467299	-1.258628	1

```
plt.figure(figsize=(10,10))
sns.scatterplot(data=df_pca, x='PCA1', y='PCA2', hue='Clusters', palette='Set2', alpha=0.8)
plt.title('Personality Clusters after PCA');
```



Tech Stocks

Collecting / Organizing Data

-Tech Stocks = ["AMZN" , "TWTR" , "GOOGL" , "FB" , "MSFT" , "AAPL" , "TSLA" , "FSR" , "NVDA" , "INTC"]

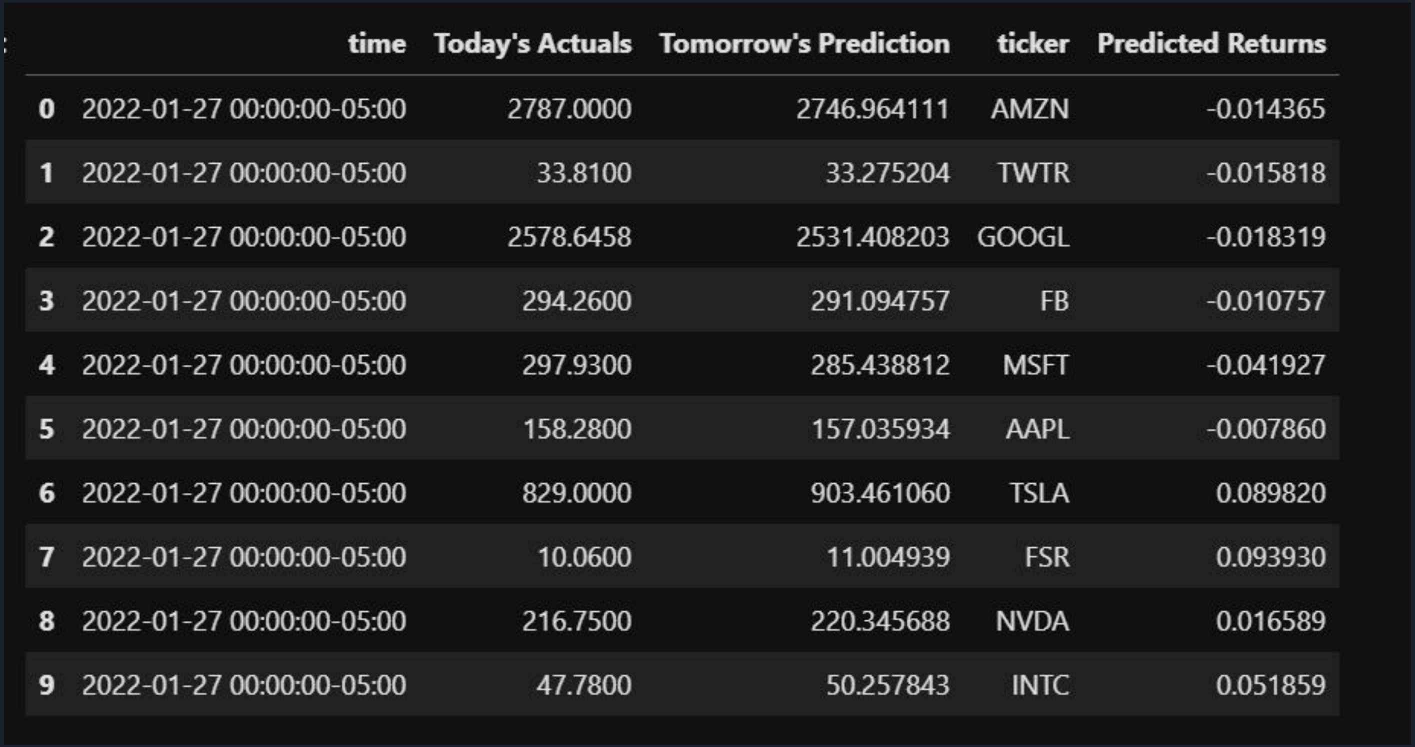
-Used Aplaca API to pull in historical Stock data for 10 Tech Stocks

-Tensorflow and Keras was used to build the recurrent neural network

- LSTM RNN model built to predict future stock prices of 10 tech stocks

Stock Prediction Results

['Predicted Returns'] = (["Tomorrow's Prediction"] - ["Today's Actuals"]) / ["Today's Actuals"]



	time	Today's Actuals	Tomorrow's Prediction	ticker	Predicted Returns
0	2022-01-27 00:00:00-05:00	2787.0000	2746.964111	AMZN	-0.014365
1	2022-01-27 00:00:00-05:00	33.8100	33.275204	TWTR	-0.015818
2	2022-01-27 00:00:00-05:00	2578.6458	2531.408203	GOOGL	-0.018319
3	2022-01-27 00:00:00-05:00	294.2600	291.094757	FB	-0.010757
4	2022-01-27 00:00:00-05:00	297.9300	285.438812	MSFT	-0.041927
5	2022-01-27 00:00:00-05:00	158.2800	157.035934	AAPL	-0.007860
6	2022-01-27 00:00:00-05:00	829.0000	903.461060	TSLA	0.089820
7	2022-01-27 00:00:00-05:00	10.0600	11.004939	FSR	0.093930
8	2022-01-27 00:00:00-05:00	216.7500	220.345688	NVDA	0.016589
9	2022-01-27 00:00:00-05:00	47.7800	50.257843	INTC	0.051859

Stock recommendation based on Personality

Tickers Clustered by Volatility:

```
split = round(len(volatility) / 3)

high = volatility[len(volatility) - split:]
mid = volatility[len(volatility) - 2 * split :len(volatility) - split]
low = volatility[:len(volatility) - 2 * split]
```

- Conscientious [Risk Tolerance: Low]
- MSFT, GOOGL, AMZN, AAPL

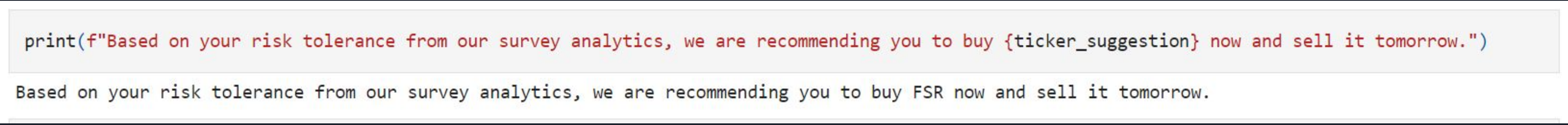
- Extraversion [Risk Tolerance: High]
- TSLA, NVDA, FSR

- Agreeableness [Risk Tolerance: Moderate]
- Open Personality [Risk Tolerance: Moderate to High]
- FB, INTC, TWTR

Selecting a stock from each data frame

```
Collection = dict()

for i in tech_stocks:

   stockframe, prediction = run_LSTM(i)

   Collection[i] = [stockframe, prediction]
```

```
mid_list = mid.index.tolist()
print(mid_list)

high_list = high.index.tolist()
print(high_list)

low_list = low.index.tolist()
print(low_list)
```



```
print(f"Based on your risk tolerance from our survey analytics, we are recommending you to buy {ticker_suggestion} now and sell it tomorrow.")
```

```
Based on your risk tolerance from our survey analytics, we are recommending you to buy FSR now and sell it tomorrow.
```

Amazon Lex Bot Creation

Our Robo Broker was created by starting with the Welcome intent

Default Slots were used such as the age of the user, name and amount of investment

Then we created a Personality Type Slot

Personality Types were then added to that Personality Type Slot

Function was created to utilize Lambda coding

The Lamba Coding was tested using age error / amount error /Negative age error test event.

Questions and minimum requirements implemented into the Bot

How old are you?

21 or older to invest

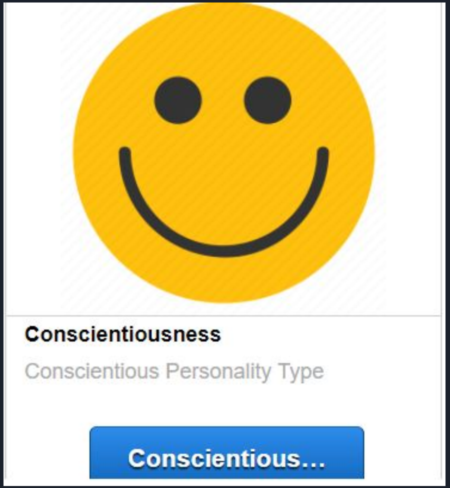
How much do you want to invest?

$1,000 or more to invest

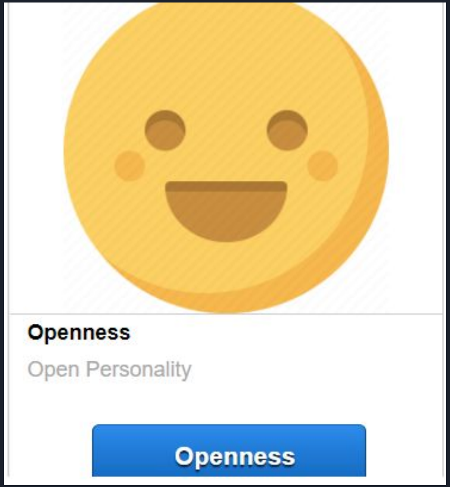
What is your name?

Gina

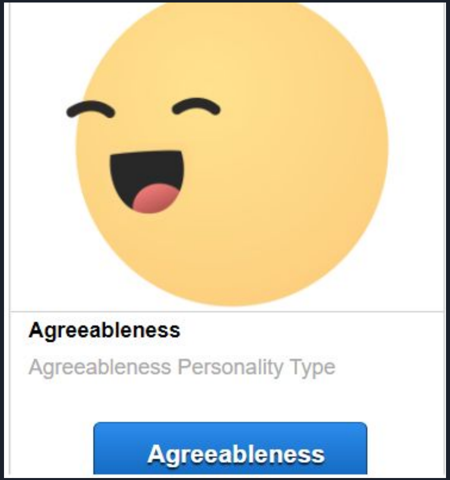
Amazon Lex Bot Personality types to choose from


Conscientiousness

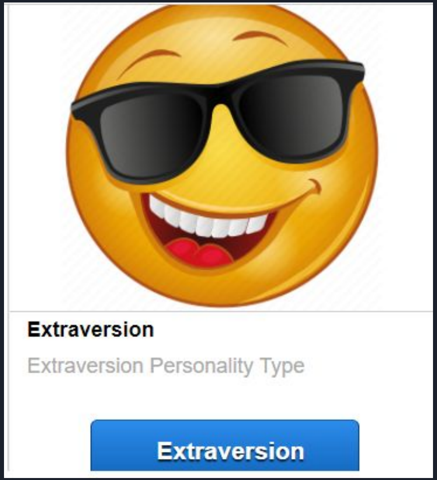
Conscientious Personality Type


Openness

Open Personality


Agreeableness

Agreeableness Personality Type


Extraversion

Extraversion Personality Type

Lambda Code

```python
    # Validate that the user's age is 21 years old
    if age is not None:
        age = parse_int(age)
        if age < 21:
            return build_validation_result(
                False,
                "Age",
                "You should be 21 years or older to use this service, "
                "please provide a different age.",
            )

    # Validate the investment amount, it should be >= 1000
    if investment_amount is not None:
        investment_amount = parse_int(
            investment_amount
        )  # Since parameters are strings it's important to cast values
        if investment_amount < 1000:
            return build_validation_result(
                False,
                "investmentAmount",
                "The minimum investment amount is 1,000 USD to use this service, "
                "please provide a greater amount.",
            )
```

Lambda Code: Stock Recommendation based on Personality Type (Risk Factor) and stock Performance (Returns)

# AAPL has High expected returns with High Risk Factor

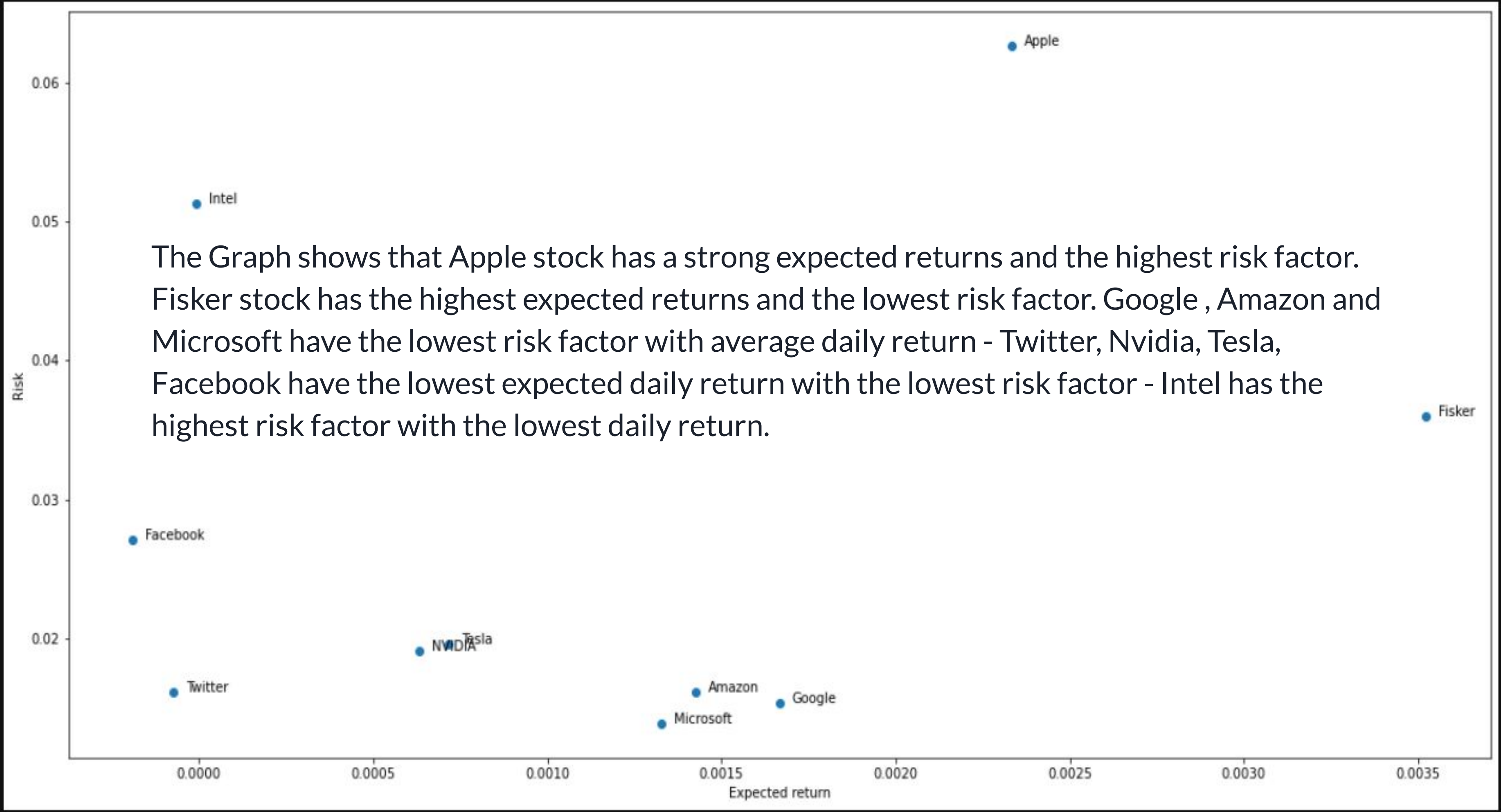
# FSR has High expected returns with Medium Risk Factor

# FB has Medium expected returns with Medium Risk Factor

# AMZN, GOOGL, MSFT have Medium expected returns with Low Risk Factor

# TSLA, TWTT & NVDIA have Low expected returns with Low Risk Factor

# INTC has Low expected returns with High Risk Factor



The Graph shows that Apple stock has a strong expected returns and the highest risk factor. Fisker stock has the highest expected returns and the lowest risk factor. Google , Amazon and Microsoft have the lowest risk factor with average daily return - Twitter, Nvidia, Tesla, Facebook have the lowest expected daily return with the lowest risk factor - Intel has the highest risk factor with the lowest daily return.

Stock Recommendation based on Personality Type (Risk Factor) and stock Performance

```
def risk(Personality_type):
    """

    """
    if Personality_type == "Conscientiousness":
        rec = "AMZN, GOOGL, MSFT, TSLA, TWTT &  NVDIA (AMZN, GOOGL, MSFT have
Medium expected returns with Low Risk Factor and TSLA, TWTT &  NVDIA have Low
expected returns with Low Risk Factor "
    elif Personality_type == "Openness":
        rec = "FB (FB has Medium expected returns with Medium Risk Factor)"
    elif Personality_type == "Agreeableness":
        rec = "FSR, (FSR has High expected returns with Medium Risk Factor) "
    elif Personality_type == "Extraversion":
        rec = "AAPL (AAPL has High expected returns with High Risk Factor) "
    else:
        rec = "INTC (INTC  has  Low expected returns with High Risk Factor)"

    return rec
```

RoboBroker - How it works

Live Demo (Video)

Conclusions

-Use more stock tickers to expand data

-Implement the personality type questions into the bot to create a scoring system to determine personality type. Have the user answer a list of questions so the bot determines personality type rather than the user.

-Go more in depth with language packages to clear up some audio clarity