Stock Market Analysis

UCB - Project 4

By: Amy Larsen, Anthony Abushacra, Karan Dogra,
Paolo Arciaga & Thotadamoole Shreenidhi

Stock Market: Basics

- **Understanding Market Participation**
  - **61%** of the American adult population is actively investing in the stock market
  - Over the past century, the S&P 500 has shown an upward trend, being up in roughly **70%** of the years

- **Why Invest?**
  - Higher earning potential
  - Beating inflation
  - Passive income
  - Ownership in a company
  - Diversification
  - Long-term wealth building

Objective

Forecast..

stock prices for specific companies using historical data and market indicators

Provide..

insights into potential future trends based on predictive modeling techniques

Important Disclosure

- **Inherent Market Risks**
  - Volatility
  - Fluctuations in prices
  - Potential loss for capital
  - Other events (War, COVID, Election)

- **Due Diligence**
  - Prior to investing, its essential to review all associated documents including prospectuses, memorandums, and any relevant terms & conditions

Industries & Companies

Our analysis focused on..

Microsoft

Google

amazon

Meta

Apple

Technology

Automotive

HONDA

TOYOTA

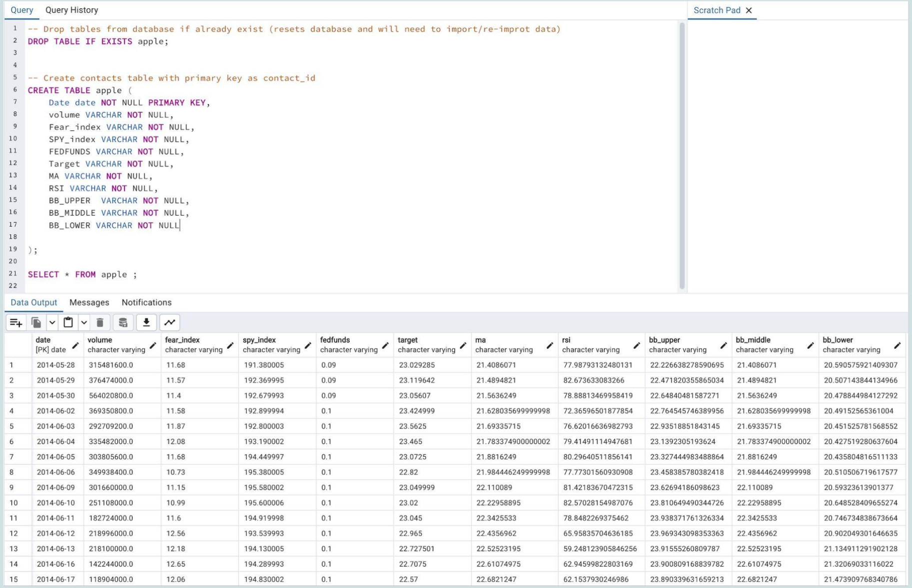
TESLA

VOLKSWAGEN

gm general motors

Database

- **Extract, Transform, & Load**
    - The data with the high, low, close, adj close, volume, Fear Index, Spy Index, and Fed Funds were extracted as CSV files from Yahoo Finance
    - The moving averages, RSI, and BB data were extracted using the finta module
    - Data was transformed to a dataframe in order to be trained & fit to the regression model
    - Data was transformed again to csv file to upload into database and create visualizations on Tableau

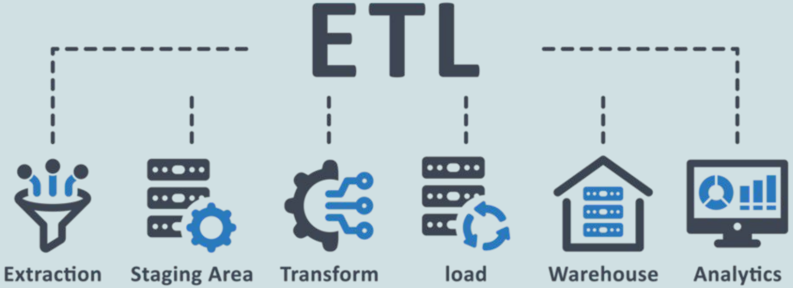


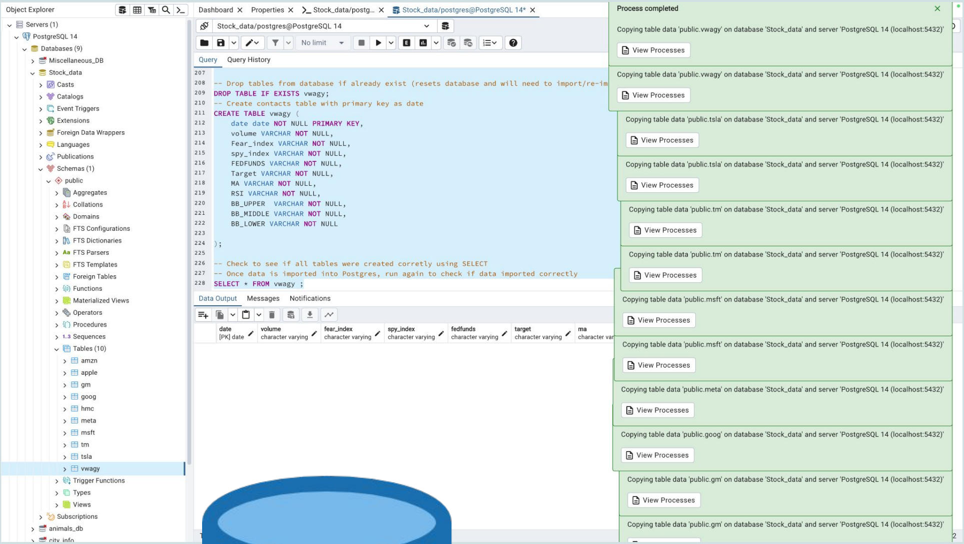
Query Query History Scratch Pad

```sql
-- Drop tables from database if already exist (resets database and will need to import/re-improt data)
DROP TABLE IF EXISTS apple;


-- Create contacts table with primary key as contact_id
CREATE TABLE apple (
    Date date NOT NULL PRIMARY KEY,
    volume VARCHAR NOT NULL,
    Fear_index VARCHAR NOT NULL,
    SPY_index VARCHAR NOT NULL,
    FEDFUNDS VARCHAR NOT NULL,
    Target VARCHAR NOT NULL,
    MA VARCHAR NOT NULL,
    RSI VARCHAR NOT NULL,
    BB_UPPER  VARCHAR NOT NULL,
    BB_MIDDLE VARCHAR NOT NULL,
    BB_LOWER VARCHAR NOT NULL

);

SELECT * FROM apple ;
```

Data Output Messages Notifications

	date [PK] date	volume character varying	fear_index character varying	spy_index character varying	fedfunds character varying	target character varying	ma character varying	rsi character varying	bb_upper character varying	bb_middle character varying	bb_lower character varying
1	2014-05-28	315481600.0	11.68	191.380005	0.09	23.029285	21.4086071	77.98793132480131	22.226638278590695	21.4086071	20.590575921409307
2	2014-05-29	376474000.0	11.57	192.369995	0.09	23.119642	21.4894821	82.673633083266	22.471820355865034	21.4894821	20.507143844134966
3	2014-05-30	564020800.0	11.4	192.679993	0.09	23.05607	21.5636249	78.88813469958419	22.64840481587271	21.5636249	20.478844984127292
4	2014-06-02	369350800.0	11.58	192.899994	0.1	23.424999	21.628035699999998	72.36596501877854	22.764545746389956	21.628035699999998	20.49152565361004
5	2014-06-03	292709200.0	11.87	192.800003	0.1	23.5625	21.69335715	76.62016636982793	22.93518851843145	21.69335715	20.451525781568552
6	2014-06-04	335482000.0	12.08	193.190002	0.1	23.465	21.783374900000002	79.41491114947681	23.1392305193624	21.783374900000002	20.427519280637604
7	2014-06-05	303805600.0	11.68	194.449997	0.1	23.0725	21.8816249	80.29640511856141	23.327444983488864	21.8816249	20.435804816511133
8	2014-06-06	349938400.0	10.73	195.380005	0.1	22.82	21.984446249999998	77.77301560930908	23.458385780382418	21.984446249999998	20.510506719617577
9	2014-06-09	301660000.0	11.15	195.580002	0.1	23.049999	22.110089	81.42183670472315	23.62694186098623	22.110089	20.59323613901377
10	2014-06-10	251108000.0	10.99	195.600006	0.1	23.02	22.22958895	82.57028154987076	23.810649490344726	22.22958895	20.648528409655274
11	2014-06-11	182724000.0	11.6	194.919998	0.1	23.045	22.3425533	78.8482269375462	23.938371761326334	22.3425533	20.746734838673664
12	2014-06-12	218996000.0	12.56	193.539993	0.1	22.965	22.4356962	65.95835704636185	23.969343098353363	22.4356962	20.902049301646635
13	2014-06-13	218100000.0	12.18	194.130005	0.1	22.727501	22.52523195	59.248123905846256	23.91555260809787	22.52523195	21.134911291902128
14	2014-06-16	142244000.0	12.65	194.289993	0.1	22.7075	22.61074975	62.94599822803169	23.900809168839782	22.61074975	21.32069033116022
15	2014-06-17	118904000.0	12.06	194.830002	0.1	22.57	22.6821247	62.1537930246986	23.890339631659213	22.6821247	21.473909768340786





Database

- **Importing Our Data**
  - After successfully training/testing our model on our companies, we imported our new prepared CSVs into pgAdmin

- **Verification**
  - After table creation, our import was verified by querying the tables

- **Data Integrity Maintenance**
  - A mechanism to reset the database was implemented by dropping existing tables before creating new ones, ensuring data integrity and consistency during re-imports.

Data Model Optimization

- **Random Forest Regressor**
  - Worked well with test data but was the model was overfitting the training data. This led to a 99% accuracy of the model.
- **Gradient Boosting Regressor**
  - Similar to Random Forest, this model lead to a very high accuracy

Data Model Implementation

- **Downloaded historical financial data**
    - Via Yahoo Finance
    - April 2014 - April 2024
- **Uploaded each dataset into Jupyter**
    - Prepared & transformed the data
- **Split data into our test and train data**
    - Trained the models

Data Model Implementation

Our 1st Model: Apple - 'AAPL'

[53]:

```python
# Filter models where both Train R-squared and Test R-squared are less than 0.96
filtered_indices = [i for i, (train_r2, test_r2) in enumerate(zip(selected_train_r2, selected_test_r2)) if train_r2 < 0.96 and test_r2 < (

# Calculate the absolute difference between train R-squared and test R-squared values for filtered models
abs_diff_r2_filtered = np.abs(np.array(selected_train_r2)[filtered_indices] - np.array(selected_test_r2)[filtered_indices])

# Find the index of the model with the smallest absolute difference among filtered models
best_model_index = filtered_indices[np.argmin(abs_diff_r2_filtered)]

# Retrieve the metrics for the best model
best_train_r2 = selected_train_r2[best_model_index]
best_test_r2 = selected_test_r2[best_model_index]
best_train_mae = selected_train_mae[best_model_index]
best_train_mse = selected_train_mse[best_model_index]
best_test_mae = selected_test_mae[best_model_index]
best_test_mse = selected_test_mse[best_model_index]

# Print metrics for the best model
print(f"Best Model - Train R-squared: {best_train_r2}, Test R-squared: {best_test_r2}, Train MAE: {best_train_mae}, Train MSE: {best_trai
```

```
Best Model - Train R-squared: 0.9379782750204122, Test R-squared: 0.9393265203901559, Train MAE: 8.751428891894557, Train MSE: 123.415157
39309779, Test MAE: 8.900921899205576, Test MSE: 124.6952598355268
```

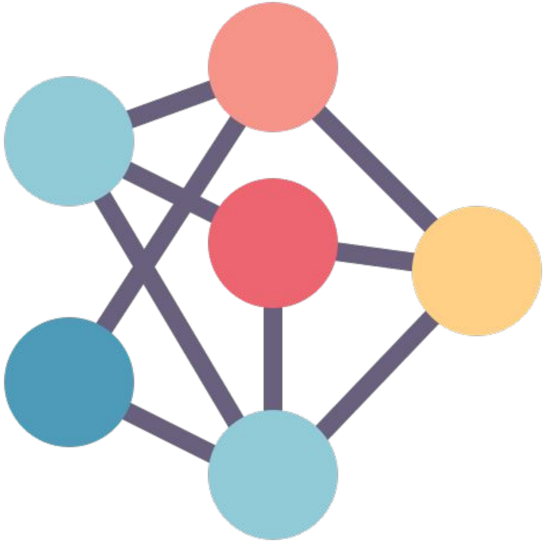
Mean Squared Error (MSE)
124.69

Root Mean Squared Error (RMSE)
11.16

R-Squared (R2)
93.9%

The MSE of 124.69 means that on average, our model's predictions for Apple's performance are about 123.69 units off from the actual values, and the high R-squared value of 93.9% tells us that it explains performance well, capturing most of the patterns in the data

Data Model Optimization



- **K-Fold Cross-Validation**
  - Evaluated the neural network model's performance

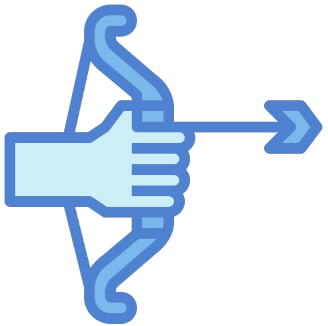
- **Compiling & Training the Model**
  - Defined a neural network model architecture with 2 hidden layers
  - Evaluation involved training the model over **50 epochs** with a smaller learning rate, assessing **R-squared scores** for both training and testing data across folds to gauge generalization ability

Data Model Optimization

Apple - 'AAPL'

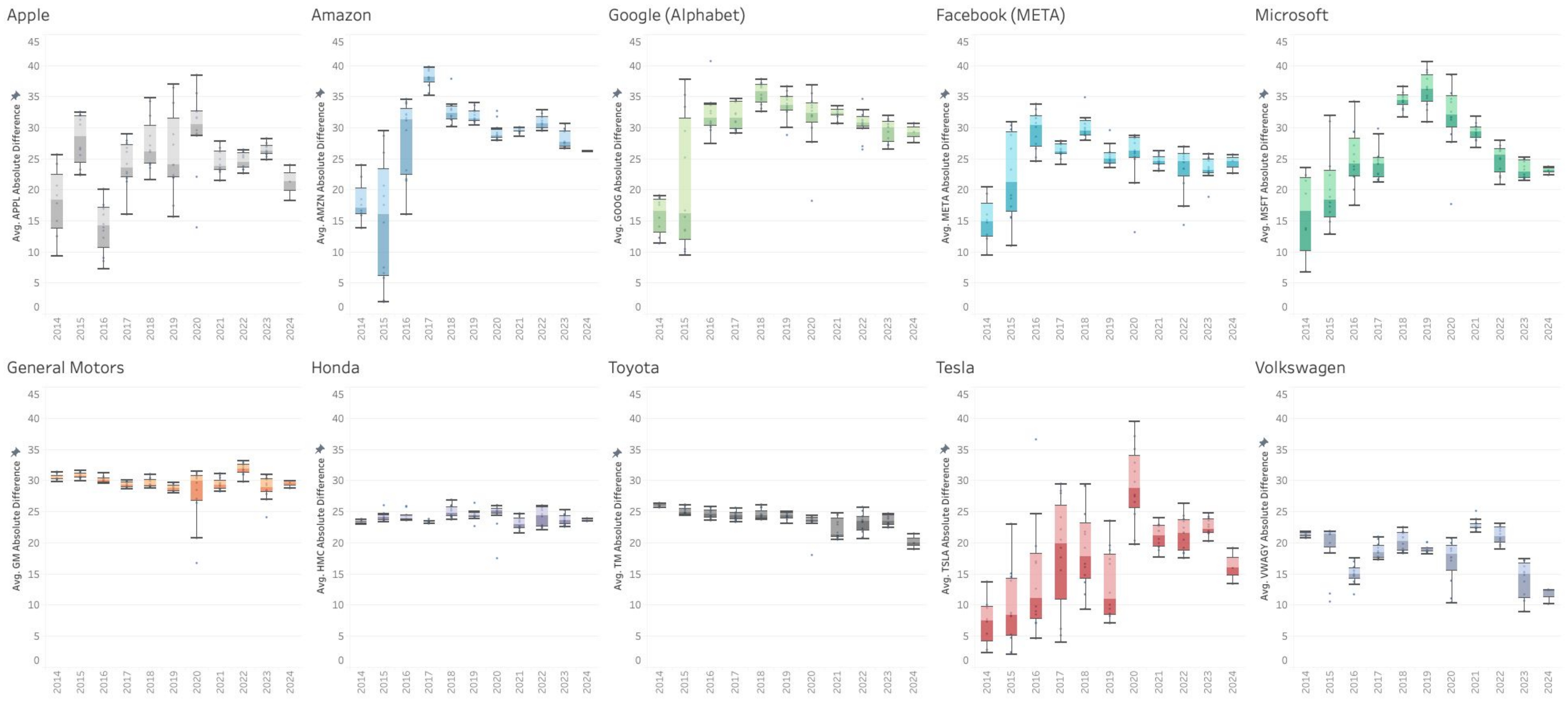
```
Model 1 – Train R-squared: 0.9615735851349676, Test R-squared: 0.9584035991161156, Train MAE: 7.168141476354025, T
rain MSE: 77.36861168528293, Test MAE: 7.034944394476862, Test MSE: 76.35683145606212
Model 2 – Train R-squared: 0.9290994571329977, Test R-squared: 0.9304370888796382, Train MAE: 8.593844473626822, T
rain MSE: 140.43590446651078, Test MAE: 8.665637809605363, Test MSE: 148.6798051905982
Model 3 – Train R-squared: 0.9478119102053203, Test R-squared: 0.9441378359672211, Train MAE: 8.438092976548905, T
rain MSE: 104.46920547531651, Test MAE: 8.537702166008373, Test MSE: 108.87611939553148
Model 4 – Train R-squared: 0.9669647385281387, Test R-squared: 0.9690153884685491, Train MAE: 6.495349016917169, T
rain MSE: 66.05083324667056, Test MAE: 6.13505515018395, Test MSE: 60.9825660603803
Model 5 – Train R-squared: 0.960908852964437, Test R-squared: 0.9567461948482872, Train MAE: 7.095985701694946, Tr
ain MSE: 78.36589854008088, Test MAE: 7.409456570652123, Test MSE: 83.17442528135216
Model 6 – Train R-squared: 0.9527591113022505, Test R-squared: 0.9440345383666274, Train MAE: 7.875742783656346, T
rain MSE: 96.54393459714595, Test MAE: 7.408290410362923, Test MSE: 86.66745906184688
Model 7 – Train R-squared: 0.9526919028849193, Test R-squared: 0.9533345877220076, Train MAE: 7.789271046483702, T
rain MSE: 93.90235823436248, Test MAE: 8.240341993653557, Test MSE: 97.8975559699576
Model 8 – Train R-squared: 0.9612718345353637, Test R-squared: 0.9648374069476484, Train MAE: 7.24067594802959, Tr
ain MSE: 76.6238012962962, Test MAE: 7.323799278727215, Test MSE: 75.78361041609146
Model 9 – Train R-squared: 0.9566785435583931, Test R-squared: 0.9584170521378743, Train MAE: 7.393871557687941, T
rain MSE: 86.11699166316636, Test MAE: 7.31254794467482, Test MSE: 86.23928420925877
Model 10 – Train R-squared: 0.9510959832772197, Test R-squared: 0.9504376608285463, Train MAE: 7.967502162618909,
Train MSE: 96.3200318823462, Test MAE: 8.617059868572177, Test MSE: 110.48203207471585
```

	Train	Test
Mean Squared Error (MSE)	93.90	97.89
Mean Absolute Error (MAE)	7.78	8.24
R-Squared (R2)	95.26%	95.33%

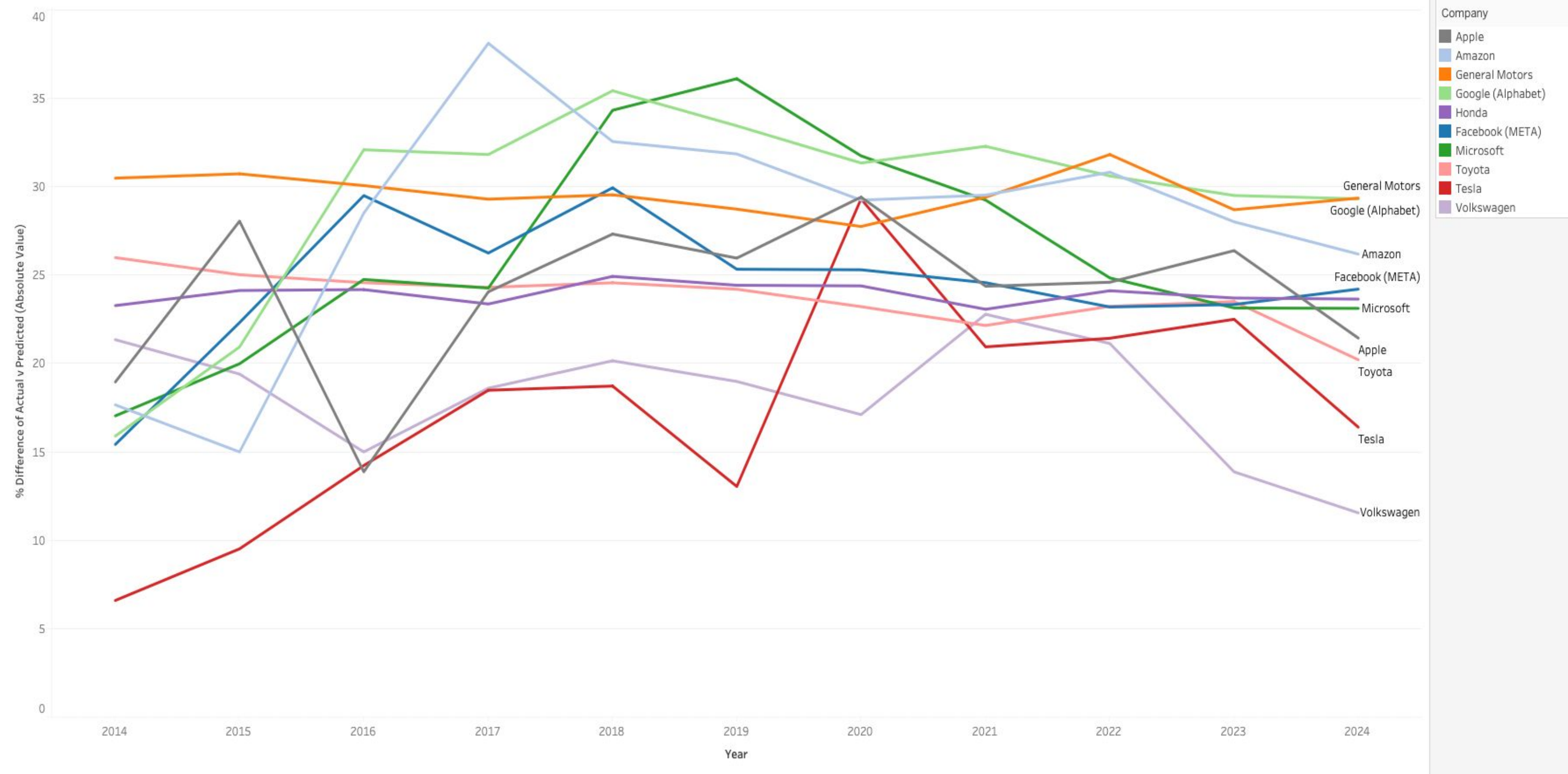


These metrics collectively provide insights into the accuracy, precision, and explanatory power of the model in predicting stock prices

Results- Yearly IQR of Absolute Value Percentage Difference (AVPD) by Month



Results- Yearly Average AVPD by Company



Final Thoughts

- **Challenges**
  - Time-Splitting Data
  - Data Overfitting- decided to drop the original features (open, high, low, close, adj. close)
  - Data Volatility- stock splits, company reports, geopolitical events

- **What's Next?**
  - Adding extra features- more sentiment analysis
  - Experimenting with new models- LSTM (possibly combined with NLP), different regressors
  - Paper Trading to test- Alpaca
  - Creating an interactive dashboard for users to experiment with

Links

- **GitHub: Stock Price Prediction**
  - Take a deeper dive into our stock market analysis, the datasets we used, and more!

- **Connect with us on LinkedIn:**
  - Amy Larsen
  - Anthony Abushacra
  - Karan Dogra
  - Paolo Arciaga
  - Thotadamoole Shreenidhi

Berkeley
UNIVERSITY OF CALIFORNIA

Thank you!