DS 5220: Supervised Machine Learning – Project Report Financial Market Analysis

Instructor: Prof Demir Utku | TA: Ashutosh Singh

Nahush Bhat, John Drohan, Harshkumar Modi

ABSTRACT

Financial time-series market data is one of the most researched categories of numerical data in the world of predictive modeling and analysis. Automated scraping libraries are freely available to allow those interested in stock market trading to download and perform data science tasks to predict the next day's stock closing price. The format and structure of these datasets are standardized, and as such contain six core features (columns). The limited availability of features leads analysts to create and generate new features and add them to the dataset to help with downstream modeling and predictive performance. The focus of our research project is to understand the importance of feature engineering for feature generation and selection, on the predictive performance of various supervised and unsupervised learning models. Primarily, we want to leverage supervised regularization techniques such as Lasso (L1 penalty) and K-Best (Wrapper) for feature generation.

I. INTRODUCTION

Since the onset of the pandemic the activity within the stock market has increased dramatically. Amateur investors will often struggle to make informed finance decisions since successful investment strategies can be locked behind pay walls. Managing a profitable portfolio for a single individual is a near impossible task that this project attempts to simplify. This would encourage a greater number of people to confidently invest their hard-earned income with a lower degree of serious risk. These investments will boost not only the economy but also the wealth of working-class citizens.

To solve this problem, it was noted that typical financial data exhibits volatility, seasonality, and trends, which requires the cumulation of multiple statistical approaches for machine learning modeling. We want to evaluate the feature engineering pipeline for time-series datasets. Financial time-series inputs are extremely hard to predict due to various market forces such as market sentiment, news, politics, and management change in addition to past ticker performance. Many existing modeling techniques in regression analysis can achieve high accuracies in prediction performance, such as ARIMA[1], sARIMA, LightGBN[2], XGboost etc. based purely on historical data and as such are not the primary scope of this project. Instead, our focus is on feature engineering and feature selection.

II. DATASET

It is extremely important to build financial prediction models using recent data. To avoid dealing with obsolete data, we decided to scrape the relevant information directly from the stock exchange platform NASDAQ using the *yfinance* library provided by Yahoo! Finance. This library allows us to extract the daily feature points based on specified date ranges. We decided to select a date range of 2 years, between 2020 and 2022. A test-train split ratio of 20:80 was adopted for all models tested in this project. The original data is stored as CSV files per individual stock ticker to ensure testing and training are performed on the same unchanged data. When it is first scraped, the CSV files include six original columns (features) (Fig 1)

Date	High	Low	Open	Close	Volume	Adj Close

Fig 1: Original Feature Set

These features are standardized across the stock market and all stock we fetch during this project follows this familiar format. The *yfinance* API is

[1] Autoregressive integrated moving average

[2] Light gradient boosted network

also very robust and ensures there are no missing or null values.

The features are described below:

1. Date – default datatype int – It contains timestamp with daily frequency
2. High – default datatype float – Represents the highest value attained by the stock ticker that day
3. Low – default datatype float – Shows the lowest trading value attained that day
4. Open – default datatype float – Final closing price during market final bell for the day
5. Volume – default datatype int – Total volume of stocks traded during the day
6. Adj Close – default datatype float – Closing price adjusted against total volume traded that day

The stock market is active for a total of 253 days per annum. Each day represents a row in our stock ticket CSV data. Since we decided to include 2 years as the range for our data to account for annual fluctuations, each stock ticket CSV will contain 506 rows along with the 6 columns mentioned above.

	A	B	C	D	E	F	G
1	Date	High	Low	Open	Close	Volume	Adj Close
2	1/12/2022	0.55	0.51	0.51	0.55	6200	0.55
3	1/13/2022	0.5	0.4	0.5	0.4	16000	0.4
4	1/14/2022	0.55	0.5	0.5	0.55	9700	0.55
5	1/18/2022	0.55	0.53	0.55	0.54	5400	0.54
6	1/19/2022	0.55	0.51	0.52	0.55	23000	0.55
7	1/20/2022	0.55	0.52	0.55	0.52	1800	0.52
8	1/21/2022	0.55	0.5	0.55	0.5	30900	0.5
9	1/24/2022	0.5	0.5	0.5	0.5	20200	0.5

Fig 2: 10 rows of data from stock ticker WFTUF

III. DATA ANALYSIS AND MODELLING

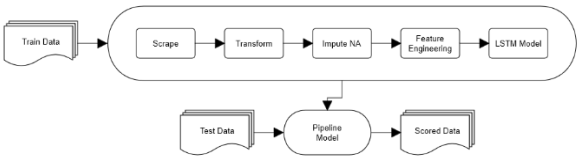

Fig 3: Full pipeline for data analysis and modelling in this project

This project follows a pipeline (Fig 3) for fast data processing and visualization. The scope wants to exhaustively investigate feature engineering techniques for stock market data. Feature selection is a process of carefully selecting an original or generated set of features to be included in a data frame before applying machine learning techniques. Typically, datasets obtained from the real world contain columns/features that do not play a huge role in predicting the target variables. Features that do not correlate with outcomes can negatively influence a prediction model's accuracy, and as such should be removed to improve performance and accuracy.

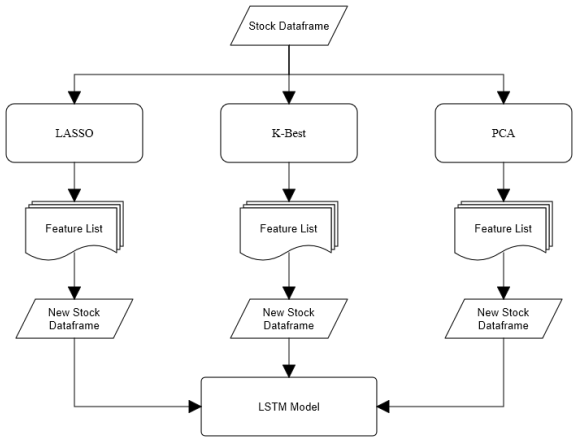

Fig 4: Feature Engineering pipeline

Anticipated Results: We expect the feature engineering pipeline to reveal features that are highly relevant for each type of time-series data described below. Stock time series has many identifying characteristics and traits which may be due to underlying factors that drive some unknown function. Instead of focusing effort on identifying all hidden factors that influence price, we estimate that a good set of carefully selected features may play a key role in downstream modeling performance. Time-series data also suffer from the overfitting problem in modeling, and the LASSO regularization technique is particularly adept at preventing overfitting. We think that this may be the key differentiator between it and the other techniques tested and we predict that while this model may be successful it could suffer from underfitting. The wrapper method K best tends to overfit when used with linear regressor as a scoring function, and since we are testing it against time series data, we anticipate that there may be a case of duplicated modeling. Finally, we expected that PCA would perform worse than the other models since we are using it for a purpose outside of pure dimensionality reduction.

In the case of stock market data, market experts and researchers have created several new features

based on mathematical manipulation of the original six. Examples of some of these are as follows:

Volume	Volatility	Trend	Seasonality
Volume-Price Trend (VPI)	Average True Range (ATR)	Simple Moving Average (SMA)	Relative Strength Index (RSI)
Negative Volume Index (NVI)	Bollinger Bands (BB)	Exponential Moving Average (EMA)	Stochastic RSI (SRSI)
Money Flow Index (MFI)	Keltner Channel (KC)	Weighted Moving Average (WMA)	Stochastic Oscillator (SO)
On-Balance Volume (OBV)	Donchian Channel (DC)	Moving Average Convergence Divergence (MACD)	Willams %R (WR)
Chaikin Money Flow (CMF)	Ulcer Index (UI)	Vortex Indicator (VI)	Rate of Change (ROC)
Ease of Movement (EMV)		Mass Index (MI)	% Price Oscillator (PPO)
Volume Weighted Average Price		Detrended Price Oscillator (DPO)	% Volume Oscillator

Fig 5: Expert features used by Market for stock analysis

The features are categorized into 4 main classes: Volume, Volatility, Trend and Seasonality. Under each class, we can find several different manual features created by market experts. These features have unique mathematical formulas associated with calculating them from the original feature-set. Under the class Volume, volume average price is generally considered to be a good feature by experts. Its formula is given by

$$\text{VWAP} = \frac{\sum \text{Price} * \text{Volume}}{\sum \text{Volume}}$$

VWAP equals the dollar value of all trading periods divided by the total trading volume for the current day. The calculation starts when trading opens and ends when it closes. Because it is good for the current trading day only, intraday periods and data are used in the calculation.

It is extremely time consuming to manually code in all the features manually, so we implemented a total of 94 new features into our dataset by calling upon the *technical analysis (TA)* library. This open-source library contains all the features used in the stock market as callable classes. We can import these new features into our data frame using a simple call command.

```
df = add_all_ta_features(
    stock, open="Open", high="High", low="Low", close="Close", volume="Volume")
```

Fig 6: code snippet of command to call ta library, all features

III.I DATA PRE-PROCESSING

As the dataset is scraped directly from the stock market, the data frame extracted is already structured. We checked for missing and null values and dropped the rows containing those. Null values can create errors while calculating the new feature set from the TA library. After importing the new features, there is a high possibility of creating new columns and rows with invalid values due to calculation errors by the automated library. Therefore, we chose to run the pre-processing step after adding all the features from TA first.

III.II EXPLORATORY DATA ANALYSIS

The first step of our EDA (Exploratory Data Analysis) phase was to identify a list of stocks for further evaluation. A useful abstraction for selecting forecasting methods is to break a time series down into systematic and unsystematic components. Since we are not choosing any existing dataset from open-source websites like Kaggle, we need to select stocks based on a certain evaluation criterion. Since a majority of the manually added features belong to 4 classes, we selected Seasonality, Trend, and Volatility as statistical criteria and evaluated a subset of 2 stocks under each one.

Seasonality: It is a characteristic of a time series in which the data experiences regular and predictable changes that recur every calendar year. Any predictable fluctuation or pattern that recurs or repeats over a one-year period is said to be seasonal. Two stocks selected under this category are:

Ticker	Name	Sector / Industry
OKE	Oneok, Inc	Oil & Natural Gas
ITW	Illinois Tool Works	Equipment Manufacturing

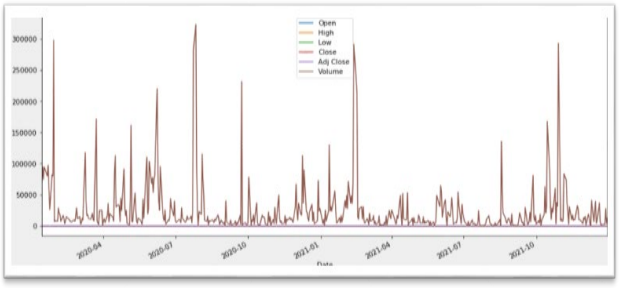

Fig 7: Seasonal pattern of ITW stock

Trend: It is a general systematic linear or (most often) nonlinear component that changes over time and does not repeat in a time series data. Stocks selected under this category are:

Ticker	Name	Sector / Industry
AAPL	Apple Inc	Software & Technology
GOOG	Google Inc	Software & Technology

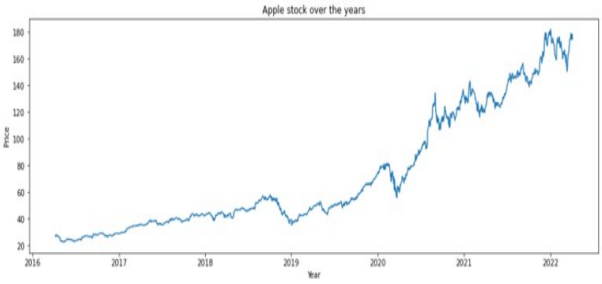

Fig 8: Clear trend of AAPL stock

Volatility: It is a non-systematic component that is nor Trend/Seasonality within the data and expresses uncorrelated movement. Stocks selected under this category are:

Ticker	Name	Sector / Industry
SPRO	Spero Therapeutics Inc	Pharmaceuticals
CHEGG	Chegg Inc	E-Learning

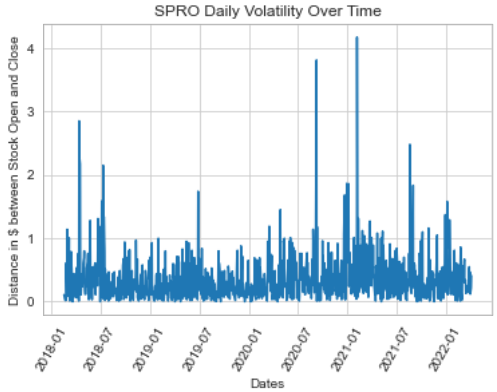

Fig 9: Volatility of SPRO stock

After selecting the list of stocks to work with, we continued our EDA by first examining the side-by-side box-plot comparisons. Stocks with strong unidirectional trends (GOOG and AAPL) tend to have a much smaller range and variance compared to stocks from the other classes. The volatile stocks exhibit a very wide range and variance.

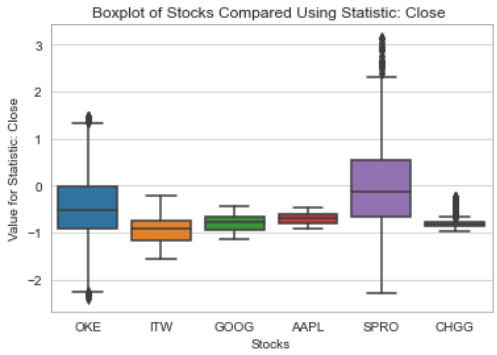

Fig 10: Box Plot comparison of all 6 stocks of distinct types.

III.III PROPOSED MODELS

Since we are evaluating feature selection techniques, the best methods under supervised learning are classified into 3 major categories: Filters, Wrappers and Embedded Learners. We intend to test one example each from the latter two and not focus on filters.

Filters	Wrappers	Embedded Learners
Generic set of methods that do not incorporate any specific ML algorithm	Evaluates against a specific ML algorithm	Embeds features during model building process. Feature selection is done by observing each iteration of model training phase
Computationally trivial	Computationally intensity proportional to number of features	Not as computationally intensive as wrappers
Low chance of overfitting	High chance of overfitting as regression is run first for feature selection then model training.	Reduces overfitting by introducing penalty terms for co-efficients of features
Example: Chi-Square test	Example: SelectKBest	Example: LASSO

Fig 11: Comparison of Feature selection methods

- **LASSO**

LASSO is abbreviated for Least Absolute Shrinkage and Selection Operator, is a method of regularization of regression loss function using an L1-penalty term added to it. This penalty is an "absolute value of magnitude" coefficient term added to the Ordinary Least Squares solution. The penalty term performs feature selection by adjusting the co-efficient weights for each feature. Those that do not affect predictive performance are automatically penalized to have 0 weight.

$$J(\theta) = \sum_{i=1}^{N} \left(h_\theta(x_i) - y^2\right) + \lambda \sum_{j=1}^{d} |\theta_j|$$

Where J(θ) is the loss function, hθ is the hypothesis, xi and yi are the input and target variables respectively, θ is the intercept and θj is the intercept at the jth iteration. Here, λ is the regularization parameter. The lasso can be rescaled so that it becomes easy to anticipate and influence the degree of shrinkage associated with a given value of λ.

Optimizing the LASSO loss function does result in some of the weights becoming zero. Thus, some of the features will be removed as a result.

- **K Best**

We wanted to compare lasso technique against another method from the supervised learning category. Although filter methods are very computationally efficient, they do not perform any machine learning, hence we selected a technique from the wrapper class of methods. For wrapper methods, the feature selection process is based on a specific machine learning algorithm that is to be applied to a particular record. It follows a greedy search approach by evaluating all possible combinations of features based on the evaluation criterion. Here, the parameter k signifies the number of features to select as a threshold for selection.

$$\rho_{xy} = \frac{\mathrm{Con}\,(r_x, r_y)}{\sigma_x \sigma_y}$$

Fig 12: Correlation formula

- **PCA**

Principal Component Analysis or PCA, is a dimensionality reduction technique that minimizes the number of features required for analysis while maximizing the amount of retained information. PCA reduces dimensions by combining the original features into principal components that can better explain the response variable. Dimensionality reduction is important for large scale machine learning models because more complex models may require hundreds of features to generate accurate results. Analyzing all these features with a complex model may prove to be computationally expensive thus limiting the usefulness of the model regardless of its accuracy. Unlike feature selection, dimensionality reduction does not remove features that lack significant correlation with the response variable. Instead, information from all the features is combined within fewer features reducing the computational cost of the model. Meaning that after a dimensionality reduction technique is applied there will likely be fewer features, each of which is better at explaining variation in the response variable than the original features of the dataset.

When performing PCA dimensionality reduction the first step is always the standardization of the data. PCA is especially sensitive to variances in the initial variables since features with larger ranges will have a more substantial impact on the outcome than those with smaller ranges. After standardization is applied to the data the mean of each feature will be set to 0 and the standard deviation to 1. Once all the features are put onto the same scale the only impact, they will have on the final component generation will be according to their variation from the mean. Shown below is the transformation applied to each feature value during scaling.

$$z = \frac{value - mean}{standard\ deviation}$$

Once the features have been standardized a covariance matrix is generated to identify correlation between each of the features. The

covariance matrix used for this purpose is symmetric and has [dxd] dimensions where d is the number of features in the original dataset. Shown below is a sample covariance matrix.

$$\begin{bmatrix} var(x) & cov(x,y) \\ cov(x,y) & var(y) \end{bmatrix}$$

The diagonal of the matrix contains the variation of each variable and each other point represents the covariance between the features corresponding to the appropriate row and column. The covariance of each feature combination can be used to identify the relationship between the two features. Specifically, the sign and magnitude of the covariance are important measurements, since the sign indicates positive or negatively correlated variables, and the magnitude explains the degree of correlation. Higher degrees of correlation indicate features that may contain redundant information and thus be superfluous to the success of the model.

The eigenvalues and eigenvectors of the covariance matrix are then calculated. The generated eigenvectors are put into descending order according to their eigenvalues, placing the newly generated features in order of their significance. The significance of the newly generated features known as principal components refers to their contribution of explained variance. Meaning that principal components that better explain the variance of the original dataset are more significant for the outcome of the model. Since the principal components are ordered by their significance the initial components explain a large portion of the overall variance. Shown below is a pair plot comparing the first three principal components based on correlation and distribution. It's clear that even just the first two principal components provide meaningful separation of the data while the third is far less efficient.

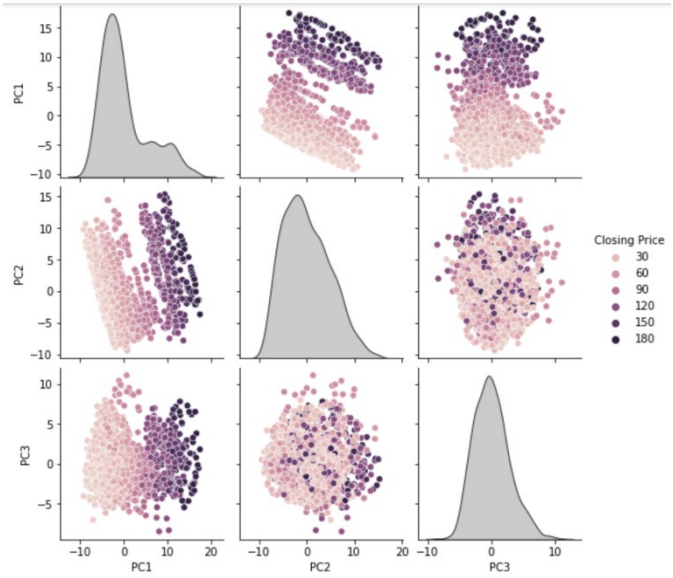

Fig 13: Pair-Plot showing PC1, PC2 and PC3 obtained

Typically, in the application of PCA the goal is to include the fewest number of components that still explain a large degree of the original variance. The degree of explained variance can vary based on the model but usually the goal is to explain at least 95% of the original data with the fewest possible components.

- **LSTM**

Long Short-Term Memory (LSTM) Network is an advanced RNN, a sequential network, that allows information to persist. It can handle the vanishing gradient problem faced by RNN. A recurrent neural network, also known as RNN is used for persistent memory. It remembers the previous information and uses it for processing the current input. The shortcoming of RNN is, they cannot remember long term dependencies due to vanishing gradients. LSTMs are explicitly designed to avoid long-term dependency problems.

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function. These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate.

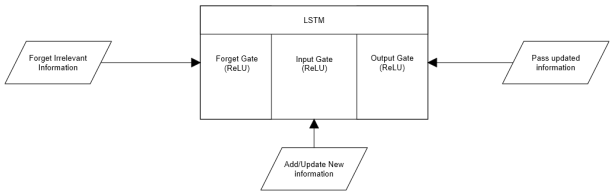

Fig 14: LSTM diagram

The Rectified Linear Activation Function (ReLU) will be used to govern our layers as it returns a value between 0 and 1.

$$f(x) = max\ (0, x)$$

Fig 15: ReLU activation function

The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn latest information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

The loss is calculated using "mean absolute error" and the optimizer is "adam" as it has shown to be the most popular optimizer with LSTMs networks.

Why Adam? Adam realizes the benefits of other techniques such as AdaGrad and RMSProp. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages. The initial value of the moving averages and beta1 and beta2 values close to 1.0 (recommended) result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.

V. IMPLEMENTATION & HYPER PARAMETER TUNING

- **LASSO**

In this project, we implemented lasso regression class from scratch without using any external libraries. The class takes in a stock data frame as input and performs L1 regularization.

Lasso regression extends Linear regression in the way that a regularization element is added to the least-squares loss function of linear regression to induce the penalty (decrease weights) against complexity (considerable number of features).

Choosing the regularization parameter is a fundamental part of lasso. Fair value is essential to the performance of the lasso since it controls the strength of shrinkage and variable selection, which, in moderation can improve both prediction accuracy and interpretability. However, if the regularization becomes too strong, important variables may be omitted and coefficients may be shrunk excessively, which can harm both predictive capacity and inferencing. Cross-validation is often used to find the regularization parameter; however, it was beyond the scope of our project, so we selected a value for λ based on information criterion scores.

An information criterion selects the estimator's regularization parameter by maximizing a model's in-sample accuracy while penalizing its effective number of parameters/degrees of freedom. Information criteria such as the Bayesian information criterion (BIC) and the Akaike information criterion (AIC) offer a computationally trivial method for finding a good λ value. An information criterion selects the estimator's regularization parameter by maximizing a model's in-sample accuracy while penalizing its effective number of parameters/degrees of freedom.

- **K Best**

We implemented K-Best with the *sklearn* package library using the *SelectKBest* function. As a scoring function, we used the *f_regressor* parameter that calls standard ordinary least squares linear regression as a machine learning model to select features. The value for k was selected as 25 to

select the top 25 features that meet the scoring function threshold.

SelectKBest will compute the F1-score statistic between each feature of x_i and y_i (assumed to be class labels). A small value will mean the feature is independent of y. A large value will mean the feature is non-randomly related to y, and so likely to provide valuable information. Only k features will be retained.

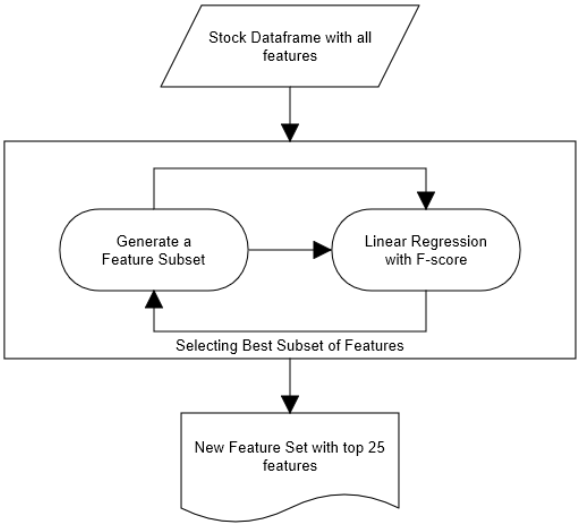

Fig 16: K-Best feature selection pipeline

- **PCA**

Since the goal of our project is to analyze the effects of different feature selection methods on the success of the LSTM model, we were primarily interested in feature significance. Typically, PCA generates a whole new set of features with fewer total dimensions that can be used instead of the original dataset. However instead of using PCA as a dimensionality reduction technique, we identified the feature significance values it utilized to create principal components.

The first step in this process was to perform the PCA using the sklearn.decomposition library. The only parameter required when running PCA with this library is the number of components that the final array will include. Fewer components require less overall computation but will lead to a lower summation of overall explained variance. When optimizing this trade off we agreed that we wanted to explain at least 95% of the variance in our original dataset. So, we started with only 1 component and checked to see exactly how much variance it could explain. Components were added iteratively until the summation of explained variance was at least 95%. Shown below is the tradeoff between components and explained variance.

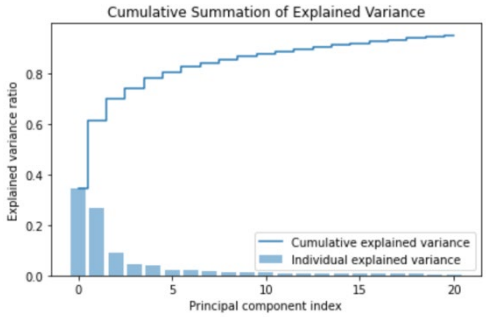

Fig 17: Cumulative summation of explained variance

Once the optimal number of principal components were generated, we needed to see exactly how much each of the original features contributed to the new components. The summation of each feature contribution towards the new components was saved into a list and ordered so that the most significant features appeared in the front. Shown below are the top 15 features and their corresponding explained variance summation scores.

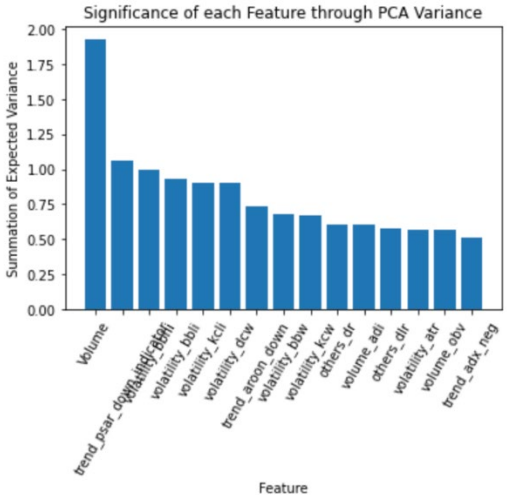

Fig 18: Significance of each feature

Once this was completed, we had a list that demonstrated the significance of each feature determined through PCA, and we simply removed every feature with a significance value below a

chosen threshold. The chosen threshold was selected to retain approximately 20 features from the original dataset. These new features were then selected to be passed along to the LSTM model.

- **LSTM**

Using the *keras* library, we define a *sequential* model. We add four layers of LSTM and three layers of dropout and one layer of dense in the network. The dropout layer is used to avoid overfitting of the data. We use the "relu" activation function as it gives only positive values as output which is what we desire.

```
#Defining the LSTM and adding layers to it
def make_model():
    model = Sequential()
    model.add(LSTM(units = 180, activation = 'relu',return_sequences = True))
    model.add(Dropout(0.2))
    model.add(LSTM(units = 60, activation = 'relu',return_sequences = True))
    model.add(Dropout(0.2))
    model.add(LSTM(units = 60, activation = 'relu',return_sequences = True))
    model.add(Dropout(0.2))
    model.add(LSTM(units = 30))
    model.add(Dense(units = 1))
    model.compile(loss = "mean_squared_error", optimizer = "adam", metrics = "mae")

    return model
```

Fig 19: Code snippet of LSTM model

To train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned. The function must also provide more sensitivity to the activation sum input and avoid easy saturation.

The architecture of this model includes four LSTM layers with the activation function as relu. We set return_sequences as True as we feed the data into the next LSTM layer having the size of output of the previous layer. Also, we included three dropout layers as it reduces the overfitting of the data. At the end we added a dense layer as we need one single output from the network. The error is set as mean squared error and the optimizer is adam, as it has shown to work best with the LSTM network.

LSTM 's was created as the solution to short-term memory. They have internal mechanisms called gates that can regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions.

An LSTM works like this; First data gets transformed into machine-readable vectors. Then the RNN processes the sequence of vectors one by one. While processing, it passes the previous hidden state to the next step of the sequence. The hidden state acts as the neural networks' memory. It holds information on previous data the network has seen before. The input and previous hidden state are combined to form a vector. That vector now has information on the current input and previous inputs. The vector goes through the relu activation, and the output is the new hidden state, or the memory of the network.

VI. RESULTS

Feature selection was performed three times for each stock, using LASSO, K-Best, and PCA, then the resulting features were used to train and test the LSTM model for each of the six stocks. The result is 18 time series plots that compare the actual values of each stock with the predicted values using each individual model. Shown below are three examples from the generated models for Apple stock trends. In the plots the orange line represents the actual stock values, and the blue line represents the predicted stock values. Along with a table that contains the R2 and Mean absolute error for each individual stock.

- **LASSO**

The lasso model obtained really good results for model precision and recall, with a tendency to approach overfitting. This is likely a result of the fact that an arbitrary value was chosen for the regularization parameter without extensive hyper parameter tuning. Reducing the variance of the model would require further testing with higher values for the regularization parameter. However, the model did yield pretty good results in terms of accuracy. CHEGG looks like it overfits the data, but the rest are realistic. These results were rather surprising since we expected the model to have an underfitting problem.

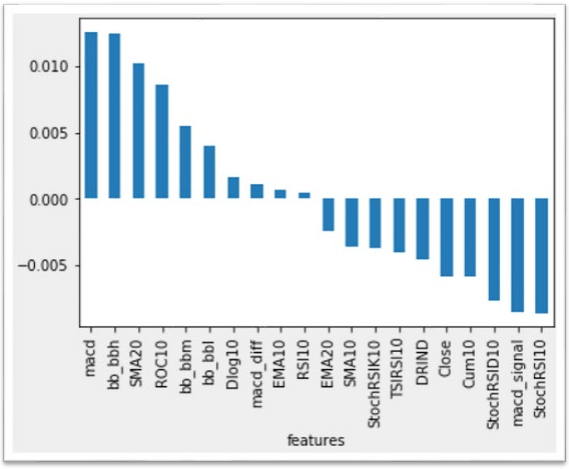

Fig 20: Feature selection by co-efficient weight

Using these set of generated features, we fed the new stock dataframe into the LSTM model for training and obtained the following scores.

Ticker	LSTM R2 Score	Mean Absolute Error
OKE	0.9893	0.0291
ITW	0.8938	0.1675
GOOG	0.7883	0.0455
AAPL	0.9426	0.1151
SPRO	0.6907	0.0263
CHEGG	0.9955	0.0026

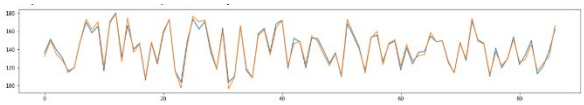

Fig 21: Predicted vs Actuals for AAPL

- **K-Best**

We observe relatively higher accuracy and performance of the model using K-best with extreme tendency to overfit the data. This is expected as K-best is using the *f_regressor* scoring functions which is basically the ordinary least squares regression model for feature selection without fitting a model. Downstream, while evaluating against the LSTM, the model will tend to overfit as a run of regression has already been performed. This result is as per our anticipated expectations, the model tends to overfit a lot and reports unrealistic accuracy scores. We would explore testing K-best using a different scoring function such as the chi-square test to prevent overfitting using an ML model. However, that makes its performance like that of filter methods.

Ticker	LSTM R2 Score	Mean Absolute Error
OKE	0.9851	0.0546
ITW	0.9001	0.0375
GOOG	0.9824	0.0490
AAPL	0.9482	0.0589
SPRO	0.8679	0.0416
CHEGG	0.9624	0.0632

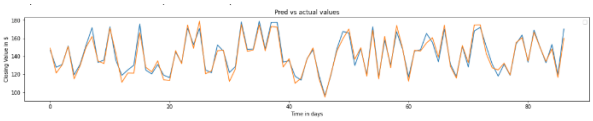

Fig 22: Predicted vs Actuals for AAPL

- **PCA**

The LSTM model trained with features selected using PCA underperformed when compared to the other feature selection methods. Typically, PCA is used for dimensionality reduction but in this project, it was repurposed for feature selection. Using PCA to accomplish a task beyond its intended function likely impacted the success of this model. Improving this model would require greater testing about how the significance of features when generating principal components corresponds to their overall predictive significance. Also, the model compared the significance of the original features over all the generated principal components, however each component explains a wildly different degree of variance. Perhaps it would have been more successful if it analyzed the feature significance contribution using only the most significant principal components. Overall, these results followed our expectations but with further analysis we believe the PCA feature selection method could eventually outperform the other employed methods.

Ticker	R2 Score	MAE
OKE	0.0082	0.2273
ITW	0.0007	0.1927
GOOG	0.0007	0.2896
AAPL	0.5021	0.1404
SPRO	0.3402	0.1390
CHEGG	-0.0201	0.2233

Fig 23: Predicted vs Actuals for AAPL

VII. CONCLUSION

Approaching the stock market can be intimidating for those who lack financial literacy, there are large financial risks associated with mistakes and everyone is looking to profit however they can. The goal of this project was to simplify the process of financial investments by highlighting stock trends over time using different feature selection methods. This would allow amateurs to make confident and informed decisions about how they invest their hard-earned income. The data for this project was obtained using the TA library which gathers information for a stock given a name and period. To ensure the model could handle different types of stock trends we analyzed stocks that were volatile, seasonal, and those with strong trends. When initially visualizing this data, we compared different features of the data set such as volume or close prices to see how they fluctuated over time. We also created box plots to compare characteristics such as close price across all the individual stocks.

We utilized two supervised methods and one unsupervised method, namely Lasso Regularization, K-Best Wrapper and PCA respectively, to build an LSTM based prediction model. This model yielded varying results for all 3 machine learning methods for feature selection. We implemented Lasso from scratch using a Python programming language, while utilizing popular machine learning packages such as *TensorFlow*, *Keras* and *sklearn* to implement the rest of the models.

The results were a fair mix of surprise and meeting expectations. We learnt that Lasso technique performs well with this type of data pipeline and use case. Its efficacy would be further improved if combined with the sister method of Ridge Regularization that utilizes an L2-penalty instead of the L1-penalty. In a field where feature selection can be performed completely manually owing to their mathematical nature, Lasso provides a reliable supervised method that can find new feature sets based on the type of stock being tested as it is exceedingly difficult for a human to write manual features for every new type of stock. PCA, on the other hand, proved remarkably unreliable with highly varying results in terms of prediction accuracy and feature selection. The method of feature selection is indirect as the model does not directly assign any co-efficient terms to the features. We opted to use a method to determine feature weights owing to their combined contribution proportions to the top 25 principal components. However, this is different from the causative effect on the target variable.

We intend to study the potential improvements that can be made to PCA feature selection technique to make it more reliable in terms of prediction performance and accuracy. During the data pre-processing, we should also evaluate each stock and measure its statistical tendencies before downstream analysis. During this project, we developed a better understanding of stock market analysis and found a new appreciation for studying the subtle differences between similar modeling techniques.

GITHUB LINK:

https://github.com/jdrohan356/Stock_Analysis_Final

IX. REFERENCES

[1] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. DOI:10.1162/neco.1997.9.8.1735. PMID 9377276

[2] Vance, Ashlee (May 15, 2018). "Quote: These powers make LSTM arguably the most commercial AI achievement, used for everything from predicting diseases to composing music". Bloomberg Business Week. Retrieved 2019-01-16.

[3] K-Best (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html)

[4] Tibshirani, Robert. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* 58, no. 1 (1996): 267–88. http://www.jstor.org/stable/2346178.

[5] PCA (https://builtin.com/data-science/step-step-explanation-principal-component-analysis)

[6] Stocks – USA: Volatility(https://www.tradingview.com/markets/stocks-usa/market-movers-most-volatile/)

[7] LSTM-time series forecasting overview:(https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/)
[8] PCA in Python with Scikit-learn:(https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60)
[9] PCA Feature Significance determination (https://stackoverflow.com/questions/22984335/recovering-features-names-of-explained-variance-ratio-in-pca-with-sklearn)