Predicting Recessions Using Machine Learning Techniques

Utilizing micro and macroeconomic data from the 2008 and 2020 recessions to predict whether the US economy is currently experiencing a recession.

I320D - Applied Machine Learning with Python
Dr. Mishra

Authors:
Michael Chen
Walt Wu
Yuning Zhang

Introduction

Background

The United States federal government is often lagged when it comes to announcing periods of economic recession, leaving plenty of room for debate among economists (Minto, 2022). For individual investors, these periods typically represent shrinkage in portfolios, lending a clear incentive to knowing whether or not the economy is in a recession before it is officially announced.

Objective

Our goal in this study is to be able to accurately predict whether or not the US economy is currently in a recession, given various micro and macroeconomic factors. Our approach begins with data collection, cleaning, and feature engineering, followed by selecting a classification model accordingly based on cross-validated accuracy metrics. We will train the models on one regime of the economy, being the 2008 recession, before taking the models out of sample into the 2020 recession. Lastly, we will attempt to optimize the best performing model using feature selection techniques before interpreting findings and drawing conclusions.

Approach and Methodology

Data Sources

We utilized two primary data sources to collect daily data:

- **YFinance** (https://pypi.org/project/yfinance/) - utilized to obtain micro-level data on banking securities. We utilized the Python package edition of YFinance in order to retrieve daily data on the Silicon Valley Bank security, including features such as the open and close price as well as volume traded. Our hypothesis behind using a banking security for an indicator of recession is that banks likely encapsulate several features of the health of the economy, such as lending or savings rates.
- **Federal Reserve Economic Data (**https://fred.stlouisfed.org/**)** - used for collecting macroeconomic indicators. The St. Louis Federal Reserve Bank publishes updated data on several macroeconomic variables. We selected some of the most popularly downloaded datasets, such as the federal bond rate, government spending, and volatility. From the FRED we also collected the label data, being the National Bureau of Economic Research's determination of time periods in recession.

We collected these variables in two time periods: one from **2005-2010**, encapsulating the 2008 recession, and one from **2020-2023**, representative of the 2020 recession.

Data Preprocessing

Having collected the security as well as macroeconomic data, we merged all of the features as well as the label into two dataframes, one for each period. Data preprocessing included dropping NaN values and duplicates. There were some inconsistencies within the FRED macroeconomic data; for instance, on days when data was not available, the FRED would fill the date with a period. These data cells were detected and forward filled with the last available point. We believe that this was most appropriate given the slow-changing nature of macroeconomic indicators and the potential information loss of dropping an entire row of data because of one missing indicator.

Feature Engineering

Along with using downloaded data, we also engineered two additional features related to our banking security that we believed would provide beneficial information for the models.

- The closing price's percentile in relation to the last 30 days of prices: we believed that this would help encapsulate the information about price changes over time, similar to how a moving average would function.
- The previous day's price information: this would add information for a longer time period than one day for the model to use without having to resort to a time-series based analysis.

Following feature engineering, we applied Min-Max Scaling on all of the features, which were entirely numerical.

After conducting data preprocessing and feature engineering, our final datasets had **1229 observations** for the first period and **786 observations** for the second period, each of these datasets having **23 features** each. Each observation represents a day, and each feature represents the various price and macroeconomic data we had collected, to be described in the following section.

One important note is that the label distribution of days that were labeled as recessions were imbalanced, warranting further discussion in this report. The following table highlights the distribution difference in days with recession vs days without:

Label	**Period 1 (2005-2010)**	**Period 2 (2020-2023)**
No Recession (0)	852	743
Recession (1)	377	43

Table 1: Distribution of Recession vs. No Recession Labels

As we elected not to resample the data to balance the label distributions, the potential ramifications of this data distribution will be discussed later.

Model Methodology

Learning Type

Since the inputs (micro and macroeconomic features) and the output (USRECD) are given, we chose to utilize a supervised learning technique in model selection.

Feature Descriptions

Again, our prediction label is the whether or not the US was in a recession on a particular day, as determined by the NBER. This was a boolean type (0 or 1), and in order to predict this label, we used the following features:

Feature Name	**Description**	**Type**
Open	The opening price of the banking security.	Numerical (Continuous)
High	The day's highest price of the banking security.	Numerical (Continuous)
Low	The day's highest price of the banking security.	Numerical (Continuous)
Close	The closing price of the banking security.	Numerical (Continuous)
Adj Close	The adjusted closing price of the banking security, accounting for things like firm actions or splits.	Numerical (Continuous)
Volume	The amount of volume of the banking security traded.	Numerical (Discrete)
Percentile_last_30	The percentile of the banking security's closing price in relation to the last 30 day's closing prices.	Numerical (Continuous)
_t1	Suffix added to the previous day's security features.	Numerical (Continuous)
T10Y2Y	The 10-Year Treasury Bond rate minus the 2-Year Bond rate.	Numerical (Continuous)
T10YIE	The 10-Year Breakeven Inflation Rate	Numerical (Continuous)
DFF	The Federal Funds Rate, i.e. government expenditures.	Numerical (Continuous)
VIXCLS	The Volatility Index, encapsulating the magnitude and speed of price changes.	Numerical (Continuous)
_SP500	Suffix added to the S&P 500's security features.	Numerical (Continuous)

Table 2: Feature Descriptions

Models

We use binary classification since we aim to predict a categorical outcome of either 0 or 1 for a given data point. To evaluate the predictive ability of the models, we calculate cross-validated average accuracy on the training dataset (**2005-2010**) and utilize a classification metric matrix on the testing dataset (**2020-2023**). The metric of 'accuracy' in the matrix serves as a measure of the model's overall predictive power, although statistics such as precision and recall provide a more in-depth understanding of the model's performance.

- For training models:
    - Average training accuracy: The average of accuracy score (the proportion of correctly classified instances out of all instances) obtained by the model during the training phase on the training dataset, which indicates how well the model is fitting the training data.
    - Average validation accuracy: The average accuracy score obtained by the model on a separate validation dataset during the training phase. The validation is unseen for the model, which means the validation accuracy is a more reliable estimate of the model's performance on new data.
- For the testing set:
    - Precision: The proportion of true positives among all predicted positives.
    - Recall: The proportion of true positives among all actual positives.
    - F1-score: A harmonic mean of precision and recall that balances both metrics.
    - Accuracy: The proportion of correctly classified instances among all instances.
    - Macro Average:The average of all classes for this metric with equal weight given to each class.
    - Weighted Average: The average of all classes, taking the number of instances in each class as a weight.

Given that our problem involves supervised learning and has a binary categorical output, we have opted to utilize the logistic regression, support vector machine, decision tree, random forest, gradient boosting, and voting models for our training and testing.

Note: All models reported below were fitted with the criterion that the features are all numerical.

Training models

To ensure that every example is present in both the training and validation data sets, allowing for multiple training and validation and averaging error through iterations, we utilize 5-fold cross-validation with the ratio of train-validation split being 4:1.

Logistic Regression

Regularization Type	Average training accuracy	Average validation accuracy
Unregularized	0.92473	0.91953
L2	0.83996	0.83543

Table 3: Logistic Regression Accuracy Metrics

The unregularized logistic regression achieved about 10% higher average accuracy scores in both training and validation compared to the L2 regularization logistic regression model. This suggests that the unregularized LR model may be more effective in accurately classifying the training dataset than the L2 LR model. The reason for this difference could be that the unregularized logistic regression has no penalty, whereas the L2 regularization logistic regression applies a penalty to prevent overfitting. This penalty tends to shrink the weights towards zero, resulting in a simpler model that may not be able to capture all the nuances in the data as effectively as the unregularized LR model.

Support Vector Machine

Kernel Type	Average training accuracy	Average validation accuracy
Linear	0.87884	0.87069
Polynomial	0.97717	0.98011

Table 4: SVM Accuracy Metrics

The results indicate that the polynomial support vector machine model outperforms the unregularized LR model by more than 5% on both training and validation data, while the linear support vector machine model performs better than the L2 LR model but worse than the unregularized LR model. We observed that the gap between training and validation accuracy of the linear SVM model is relatively larger than that of the LR model, indicating a potential overfitting issue. In contrast to linear SVM, the poly SVM model achieves almost perfect accuracy on both training and validation datasets, which can be attributed to the higher complexity and flexibility of the polynomial kernel compared to the linear kernel.

Decision Tree

Average training accuracy	Average validation accuracy
1.0	0.98643

Table 5: Decision Tree Accuracy Metrics

The decision tree model performs with perfect accuracy on the training data and a pretty high accuracy on the validation data, surpassing the accuracy of the poly SVM model. This is because the decision tree model can capture complex relationships between input variables and the target variable, as it splits the data based on the feature that provides the most information gain. The tree-based structure of the model may have helped it compensate for any multicollinearity in the features. Thus, these more complex tree-based models can be a good choice.

Random Forest

Average training accuracy	Average validation accuracy
1.0	0.99457

Table 6: Random Forest Accuracy Metrics

Random forest is an ensemble learning technique that builds multiple decision trees and combines their predictions to improve the model's accuracy and reduce overfitting. This may be why the random forest model has a better performance on the validation data than the decision tree model.

Gradient Boosting

Average training accuracy	Average validation accuracy
1.0	0.99367

Table 7: Gradient Boosting Accuracy Metrics

The Gradient Boosting also achieves perfect accuracy on the training data and a high accuracy on the validation data. Gradient Boosting is designed to correct the errors made by the previous model and eventually build a strong model. This approach may result in a more complex model than a random tree, making it more susceptible to overfitting if the coefficient or parameter selection is not appropriate. As a result, the average validation accuracy of Gradient Boosting is slightly lower than that of the random forest model.

Voting

Average training accuracy	Average validation accuracy
0.96406	0.95027

Table 8: Voting Accuracy Metrics

We also tested the voting model, which is a model that combines predictions from multiple models. The training accuracy of the voting model is higher than its validation accuracy, suggesting a potential for slight overfitting to the training data. Despite this, the validation accuracy of the model is still relatively high, indicating that the model is performing well on unseen data.

Testing models

As the random forest model exhibits the highest validation accuracy among all the models, we have selected it as our best model and will now evaluate its performance on the testing dataset **(2020 - 2023)**.

	Precision	Recall	f1-score	Support
0	1.00	0.37	0.54	743
1	0.08	1.00	0.15	43
Accuracy			0.40	786
Macro AVG	0.54	0.68	0.35	786
Weighted AVG	0.95	0.40	0.52	786

Table 9: Test Set Classification Metric Matrix

The test accuracy shows the random tree model only correctly classifies 40% of the test data, which is an overall poor performance. Both weighted average and macro average f1-scores are low, indicating that the model is performing poor across both classes. Class 0 has a high precision and low recall, while class 1 suffers from the opposite issue. This indicates that the model is likely predicting class 1 (recession) too often. Therefore, in an attempt to improve performance, feature selection will be applied to the best model, which is the random tree model.

Feature Selection

After finding out which model has the highest accuracy in predicting the recessions, we then proceeded to perform feature selection to exclude the features that are less important. Feature selection is a crucial step in data analysis and modeling, as it helps to identify the most relevant and informative features in a dataset while eliminating the noise and redundant features. By selecting the most important features, we can reduce the dimensionality of the data, which can improve the accuracy and performance of our model. Additionally, feature selection can help to address the issue of overfitting, which in our model the prediction fits too closely to the training data and fails to generalize well to the 2020-2023 data. Therefore, by doing feature selection, we can improve the efficiency, interpretability, and generalizability of our models, and make better decisions based on the data. As some of the feature selection techniques we implemented, such as correlation based filtering, were not entirely relevant to our final model, we have included extra details in Appendix II.

Ablation Test:

We first conducted an ablation test to evaluate the model prediction performance to determine the impact of each feature by removing or disabling specific components, and then examine the accuracy drop after each feature being removed.

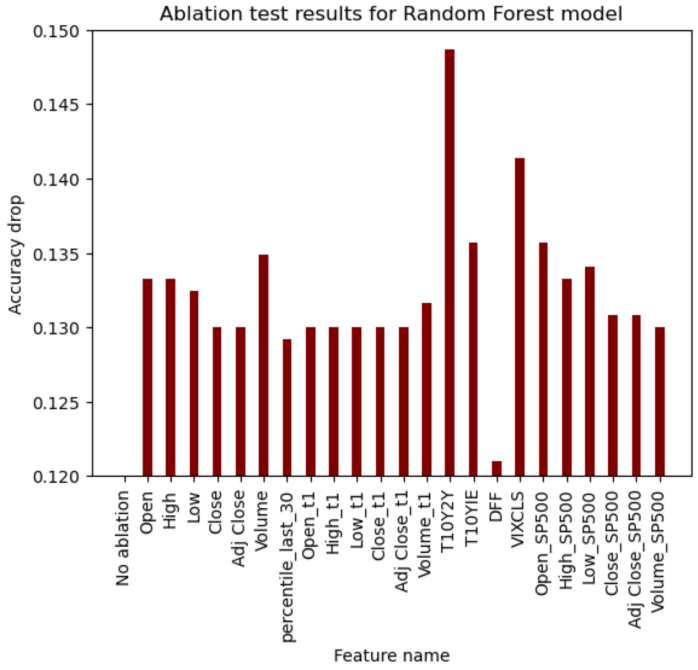


Figure 1: Ablation Test Results

After the ablation test we can tell from the results that all the features had an approximately 12% to 14% accuracy drop after being removed, which indicates that they are almost equally important. The following table describes the top and bottom 5 features in the ablation testing:

Top 5 Features	**Bottom 5 Features**
'T10Y2Y': 0.148	'DFF': 0.121
'VIXCLS': 0.141	'percentile_last_30': 0.129
'T10YIE': 0.135	'Adj Close_t1': 0.129
'Open_SP500': 0.135	'Close_t1': 0.129
'Volume': 0.134	'Low_t1': 0.129

Table 10: Top 5 Most and Least Influential Features in an Ablation Test

Rather than immediately deleting features at this stage, we were instead seeking an overall idea of what features were more important than others. Because the features are all relatively equally important, we are not able to justify removing any one of them just by looking at the ablation test.

Variance thresholding:
We then conducted a variance thresholding method for features selection. The idea is that if a feature exhibits low variance, it is not very informative. We set our variance threshold to 0.02, indicating that the method should help exclude the features that lack diverse values..

After applying the method we got the result of excluding the following 4 features:

1. High
2. Volume
3. High_t1
4. Volume_t1

The following tables represent the classification metric matrices before and after dropping features:

	Precision	Recall	f1-score	Support
0	1.00	0.37	0.54	743
1	0.08	1.00	0.15	43
Accuracy			0.40	786
Macro AVG	0.54	0.68	0.35	786
Weighted AVG	0.95	0.40	0.52	786

Before Dropping Features

	Precision	Recall	f1-score	Support
0	1.00	0.41	0.58	743
1	0.09	1.00	0.16	43
Accuracy			0.44	786
Macro AVG	0.54	0.70	0.37	786
Weighted AVG	0.95	0.44	0.56	786

After Dropping Features

Table 11: Classification Metrics before and after dropping features

We observed a slight improvement on all the f1-Scores and recalls and a 0.01 increase in precision for the recession class. This indicates a marginally better model after dropping low variance features.

Findings and Analysis

Unfortunately, even following a feature selection process, overall model performance out of sample is still low. The drop in overall accuracy when moving from the training set to the out of sample testing set indicates overfitting on the training set's data and label distributions. The high precision of the no recession class (0) is countered by the low precision of the recession labels. While the recession class (1) does have a recall of 1, meaning that it was able to find all of the cases that were in recession, the low precision means that it makes many false positive predictions. A high-level interpretation of this is that the model takes a very negative stance on the economy, meaning that it predicts that we are in a recession far more than we are actually in. However, for the non-recession class, it has a high precision and low recall, revealing that while it does miss some of the non-recession cases, when it does indeed predict recession, it is accurate. Thus, when the model predicts recession, it should be viewed quite skeptically, but when it predicts that we are not in a recession, it can be relatively trusted.

This precision-recall dynamic between classes in the test dataset can potentially be attributed to the imbalance of labels within the data. In the first period, the recession periods make up for approximately 45% of the set, while in the test dataset, only about 5% of the days are considered as in recession. This means that in the training set, the model may have learned to predict days as in recession much more liberally. When taken out of sample into the second time period's data set, this resulted in it predicting much more days as recession as there truly were.

In further taking the model out of sample, we attempted to have the model predict whether or not we were in a recession for an even more recent day:

Date	Open	High	Remaining Features	Prediction
4/25/2023	0.72	0.75	...	0 (No Recession)

Table 12: Recent Prediction Results

Considering the precision-recall statistics for the testing set (high precision but low recall for class 0), we can be relatively confident that we are not in a period that the NBER would label as a recession (this is debatable to current economists and none of this paper should be taken as financial advice).

In practice, this model would only be useful in niche cases, considering that it takes a very negative stance on the economy and predicts that we are in a recession too often. This would be very alarming to personal investors, as if it were taken as investment advice, it could potentially incite panic. However, the fact that it does have a high precision for the non-recession class means that it could potentially reassure scared investors when it does predict that we are not in a recession.

Considerations and Further Research

The extreme drop in overall model accuracy once taken out of sample is indicative of overfitting on the training set, which can partly be attributed to the data distribution discussed as well as the model selected. This reveals an overall consideration for the use of financial data: factors that work well in one regime of the economy may not continue to apply in the future. Using machine learning models to describe financial situations requires constant adaptation to new information. Because complex models are more prone to overfitting, it is important for all individuals using models for themselves to be cognisant of how models were trained and how they are reaching their conclusions.

There are several additional techniques and variables that should be considered when attempting future research on this model:

- **Resampling methods**: because the periods of recession were much more plentiful in the training dataset, the model likely learned to predict recessions more often than it should have. Techniques such as upsampling the non-recession periods or downsampling the recession periods to more closely match more recent recession can be useful in making the model more accurate in predicting out of sample.
- **Feature selection:** including a higher variety of features, such as technical indicators along with the current features selected, may provide some additional information about the state of the US economy that we are currently not able to capture. Furthermore, applying feature selection techniques, such as LASSO regularization, may provide insight into the most important features even when using a model that does not support regularization techniques.

- **Hyperparameter tuning:** adjusting parameters, such as the number of trees or pruning measures in a random forest model, may improve performance.

Conclusion

Despite most of the classification models having high performance on the training set of 2005-2010, once taken out of sample, the chosen model performance significantly dropped, indicating that the model was overfit on the training data. While the model performs well if it predicts days as not in recession, it labels too many days to be in a recession. The precision-recall dynamic between the classification labels points to the distribution of labels as a problem to address in future optimization. Thus, we believe that while our current model is not ready to ship, it does serve as a basis for further research regarding the use of machine learning techniques in predicting recessions in the economy. The economic markets are a constantly shifting organism driven by human behavior, and so constant up-to-date research and adaptation is required to grasp a higher understanding of them.

Appendices

Appendix I: Q&A

There were two questions raised regarding the report: the first was for a description of the distribution of the labels, which we have provided in Table 1.

The second question raised was regarding using LASSO feature selection; whether or not it was possible to apply features selected by LASSO on another model into the random forest. The results are as below:

We chose to apply the lasso regularization to the linearSVC model first then apply the features being selected to the random forest model, which are the following features:

```
Index(['T10Y2Y', 'DFF', 'VIXCLS', 'High_SP500', 'Volume_SP500'], dtype='object')
```

After applying the 5 features to create a new reduced column dataset, we then trained a new random forest model based on the 5 features and received the following classification report after the new predictions:

	Precision	Recall	f1-score	Support
0	1.00	0.37	0.54	743
1	0.08	1.00	0.15	43
Accuracy			0.40	786
Macro AVG	0.54	0.68	0.35	786
Weighted AVG	0.95	0.40	0.52	786

Before LASSO Selection

	Precision	Recall	f1-score	Support
0	1.00	0.42	0.59	743
1	0.09	1.00	0.17	43
Accuracy			0.45	786
Macro AVG	0.55	0.71	0.38	786
Weighted AVG	0.95	0.45	0.57	786

After Lasso Selection

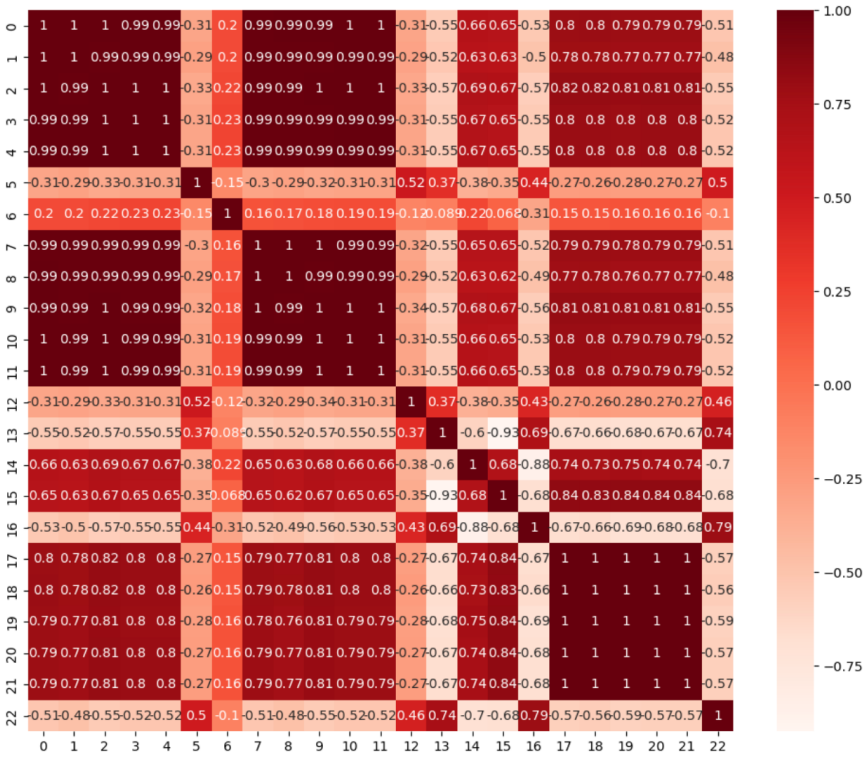
Once again, we observed a slight improvement on all the f1-Scores and recalls. This reveals that despite the LASSO regularization dropping most of the features in our model, we were still able to get an improved model. However, it appears that simply feature reduction does not solve the overfitting issues present structurally within the data.

Appendix II: Further Feature Selection Techniques

These feature selection techniques reveal interesting traits about the dataset, but as they were not directly relevant to the random forest model we implemented, we have elected to include them here.

Correlation based filtering:

We then conducted a correlation based filtering method hoping that we can find out some features to be excluded. The idea here is that if two features are extremely correlated, we can get rid of one of them, thereby reducing feature redundancy. However, since all the features in our dataset have at least one extremely correlated feature, this method is not applicable in our case.



Variance Filtering:

The variance for each of the features can be seen described below:

```
['Open' 'High' 'Low' 'Close' 'Adj Close' 'Volume' 'percentile_last_30'
 'Open_t1' 'High_t1' 'Low_t1' 'Close_t1' 'Adj Close_t1' 'Volume_t1'
 'T10Y2Y' 'T10YIE' 'DFF' 'VIXCLS' 'Open_SP500' 'High_SP500' 'Low_SP500'
 'Close_SP500' 'Adj Close_SP500' 'Volume_SP500']
[0.02608834 0.0178584  0.03941526 0.03015916 0.03015916 0.00318741
 0.1259201  0.02601272 0.01783327 0.03930229 0.03012691 0.03012691
 0.00314783 0.10055797 0.03969996 0.13436202 0.03091459 0.05319098
 0.05284851 0.05371838 0.05284739 0.05284739 0.02566019]
Selected features: ['Open' 'Low' 'Close' 'Adj Close' 'percentile_last_30' 'Open_t1' 'Low_t1'
 'Close_t1' 'Adj Close_t1' 'T10Y2Y' 'T10YIE' 'DFF' 'VIXCLS' 'Open_SP500'
 'High_SP500' 'Low_SP500' 'Close_SP500' 'Adj Close_SP500' 'Volume_SP500']
```

Chi-squared test:

Chi-squared test is a commonly used mechanism to accept or reject whether two variable distributions are equal or not. If they are very different, the chi-squared score will be significantly different from the ideal value, suggesting that the features and labels are completely independent. In this case, we can remove the feature.

```
Ranking based on chi-squared test
Printing ranked features based on chi-squared value
lower p-value indicates feature and label are dependent)
Feature T10Y2Y : p-value 4.3627939440908385e-26
Feature VIXCLS : p-value 1.9461152821496286e-23
Feature DFF : p-value 3.3043149154799949e-22
Feature Volume_SP500 : p-value 1.9031100642533394e-12
Feature Low_SP500 : p-value 6.838464653194965e-06
Feature Adj Close_SP500 : p-value 1.3273743881994899e-05
Feature Close_SP500 : p-value 1.3273743881994899e-05
Feature Open_SP500 : p-value 1.4524523267619493e-05
Feature T10YIE : p-value 1.9094177412174084e-05
Feature High_SP500 : p-value 2.219390325670091e-05
Feature Low : p-value 0.0008271022075340096
Feature Low_t1 : p-value 0.0008985386189076504
Feature Adj Close : p-value 0.0033857296856412084
Feature Close : p-value 0.0033857296856412084
Feature Volume : p-value 0.0036282505377850474
Feature Adj Close_t1 : p-value 0.003798176528950475
Feature Close_t1 : p-value 0.003798176528950475
Feature Volume_t1 : p-value 0.004928960091348383
Feature Open : p-value 0.005417259892815673
Feature Open_t1 : p-value 0.005692332577632015
Feature percentile_last_30 : p-value 0.017883854917086495
Feature High : p-value 0.0189757391748305
Feature High_t1 : p-value 0.02018541828356749
```

In result, we found that all the features have a p-value greater than 0.05, which rejects the null hypothesis that there is no significant association between the two categorical variables. In other words, the observed frequencies for each combination of categories are equal to the expected frequencies assuming the variables are independent.

Lasso Regularization:
LASSO (Least Absolute Shrinkage and Selection Operator) is a regularization technique used in linear regression to prevent overfitting and improve the model's generalization performance. However, the Lasso regularization adds a penalty term to the cost function of the linear regression model that limits the size of the regression coefficients which our random forest model does not have. Therefore, the lasso regularization does not apply to our best model.

Replicability

The data, code, and documentation necessary for replicating the results of this report can be found at this GitHub Repository: https://github.com/alchemicHen/i320D_ML_Final

References

Board of Governors of the Federal Reserve System (US), Federal Funds Effective Rate [DFF], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/DFF, April 29, 2023.

Chicago Board Options Exchange, CBOE Volatility Index: VIX [VIXCLS], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/VIXCLS, April 29, 2023.

Minto, Rob. (2022). *Calling a Recession: How Long Does It Take?*. Newsweek. Retrieved from https://www.newsweek.com/calling-recession-how-long-does-it-take-1735121, April 29, 2023.

Federal Reserve Bank of St. Louis, 10-Year Treasury Constant Maturity Minus 2-Year Treasury Constant Maturity [T10Y2Y], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/T10Y2Y, April 29, 2023.

Federal Reserve Bank of St. Louis, 10-Year Breakeven Inflation Rate [T10YIE], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/T10YIE, April 29, 2023.

Federal Reserve Bank of St. Louis, NBER based Recession Indicators for the United States from the Period following the Peak through the Trough [USRECD], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/USRECD, April 28, 2023.

YFinance. (n.d.) *Download market data from Yahoo! Finance API*. https://pypi.org/project/yfinance/