Classification & Regression Tasks in Financial Contexts: Credit Score Classification and Economic Cycle Regression

Hui-Erh Chai, Richie Ma, & Emma Mayes

IE 517: Machine Learning in Finance

Section A/AO, Fall 2021

CHAPTER 1: Credit Score Classification

1.1 Introduction & Exploratory Data Analysis (EDA)

The dataset used, MLF_GP1_CreditScore, includes 1,700 observations of 26 financial and accounting metrics changes for a set of firms in several different industries. The class label of the first dataset is the Moody's credit rating designated to the firm in a given quarter, which is assigned in a range from Aaa, representing the highest quality, down to C, which is the lowest quality. Ratings that are Baa3 or higher are considered investment grade, while ratings Ba1 and lower are considered not investment grade due to low security. Each of the ratings in the dataset are considered if they are "Investment Grade" or not based on their rating, which is represented in the dataset as a binary variable. If a financial asset is assigned 0, or is not investment grade, this asset may not be held in certain institutional portfolios (e.g., pension plans). The specific interest in this classification is in differentiating between the upper and lower B-grade bonds (i.e., between Ba1 and Baa3), as upper B-grade bonds are considered secure enough to be investment grade, but lower B-grade bonds are considered riskier. Two classification problems are going to be solved based on the first dataset. The first is a multiclass approach that aims to classify the letter of the Moody's credit rating (where the target variable is "Rating"*)* and the second is a binary approach that aims to categorize simply whether or not a bond is investment grade (where the target variable is "InvGrd"*)*.

The statistical description of the credit score dataset is shown in Table 1.1. Considering the dataset includes 26 variables, we only display 8 variables randomly due to the space constraint, including the target variable for the binary approach, "InvGrd".

Table 1.1: *Statistical description of 8 variables randomly selected from dataset*

	Total Liquidity	**Current Liquidity**	**Current Liabilities**	**EPS Bef. Extras**	**PE**	**ROA**	**ROE**	**InvGrd**
count	1700.000	1700.000	1700.000	1700.000	1700.000	1700.000	1700.000	1700.000
mean	-0.856	0.436	0.073	0.032	0.498	0.019	-0.218	0.757
std	22.927	1.904	0.266	6.152	12.103	14.594	15.389	0.429
min	-502.000	-0.994	-0.685	-96.250	-59.795	-305.462	-373.837	0.000
25%	-0.857	-0.227	-0.073	-0.153	-0.294	-0.208	-0.234	1.000
50%	-0.229	0.040	0.042	0.066	-0.040	-0.009	-0.020	1.000
75%	0.513	0.416	0.161	0.236	0.169	0.156	0.202	1.000
max	280.139	34.372	4.194	187.000	381.243	474.847	343.145	1.000

Next, we consider 25 variables, excluding our target variables "Rating" and "InvGrd", to conduct our exploratory data analysis. We standardize our feature data since the unit of each variable is different. Then, we display the distribution of each feature with a boxplot, shown in Figure 1.1. Figure 1.1 illustrates the dispersion of each variable is really large and there are many outliers in each variable.

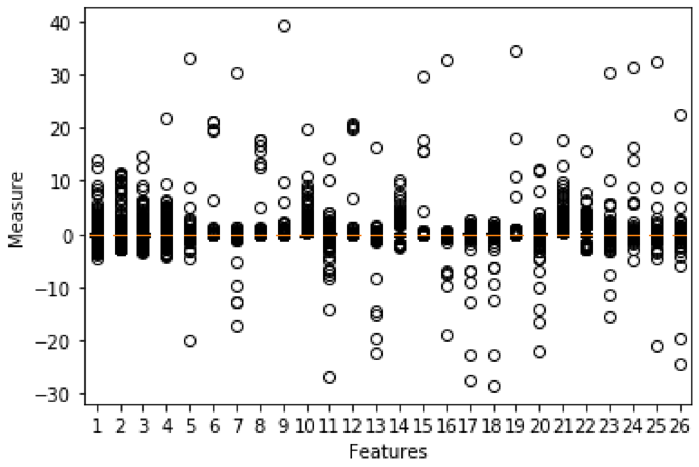


Figure 1.1: *The distribution of each of the 26 variables in the dataset*

With regard to one target variable, the Moody's rating, we plot its distribution by histogram, which could be helpful for us to know how the bonds in our dataset are rated. The results are shown in Figure 1.2. From Figure 1.2, one can find that most bonds are rated as "Baa2", which indicates that these bonds are with speculative fundamentals while the security of future payments is only moderate. The histogram also illustrates class imbalance within our dataset.

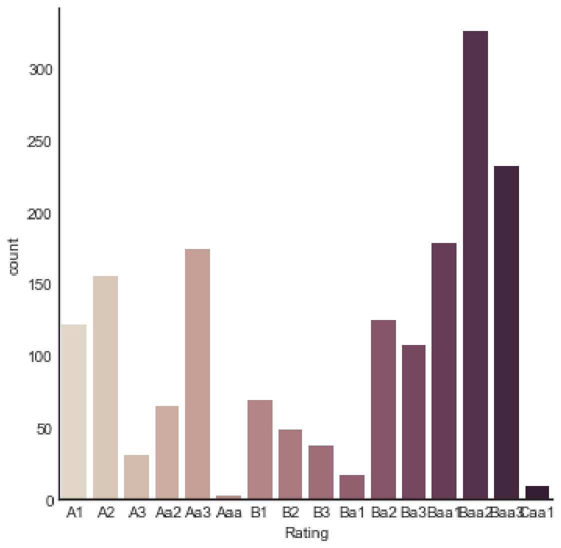


Figure 1.2: *The histogram of Moody's Credit Rating Scores for the dataset*

Finally, we plot the correlation coefficients among each feature and with the target variable using Pearson's correlation coefficient in the heatmap shown in Figure 1.3. As shown in Figure 1.3, we find most correlations between pairs of variables are generally weak and below a Pearson's correlation coefficient of 0.5, as indicated by the majority of cooler colors in the matrix.

When plotting the feature space in a correlation matrix heat map, there is noticeable correlation between certain features. It indicates a need for feature extraction and/or selection to help reduce the large dimension of the dataset.

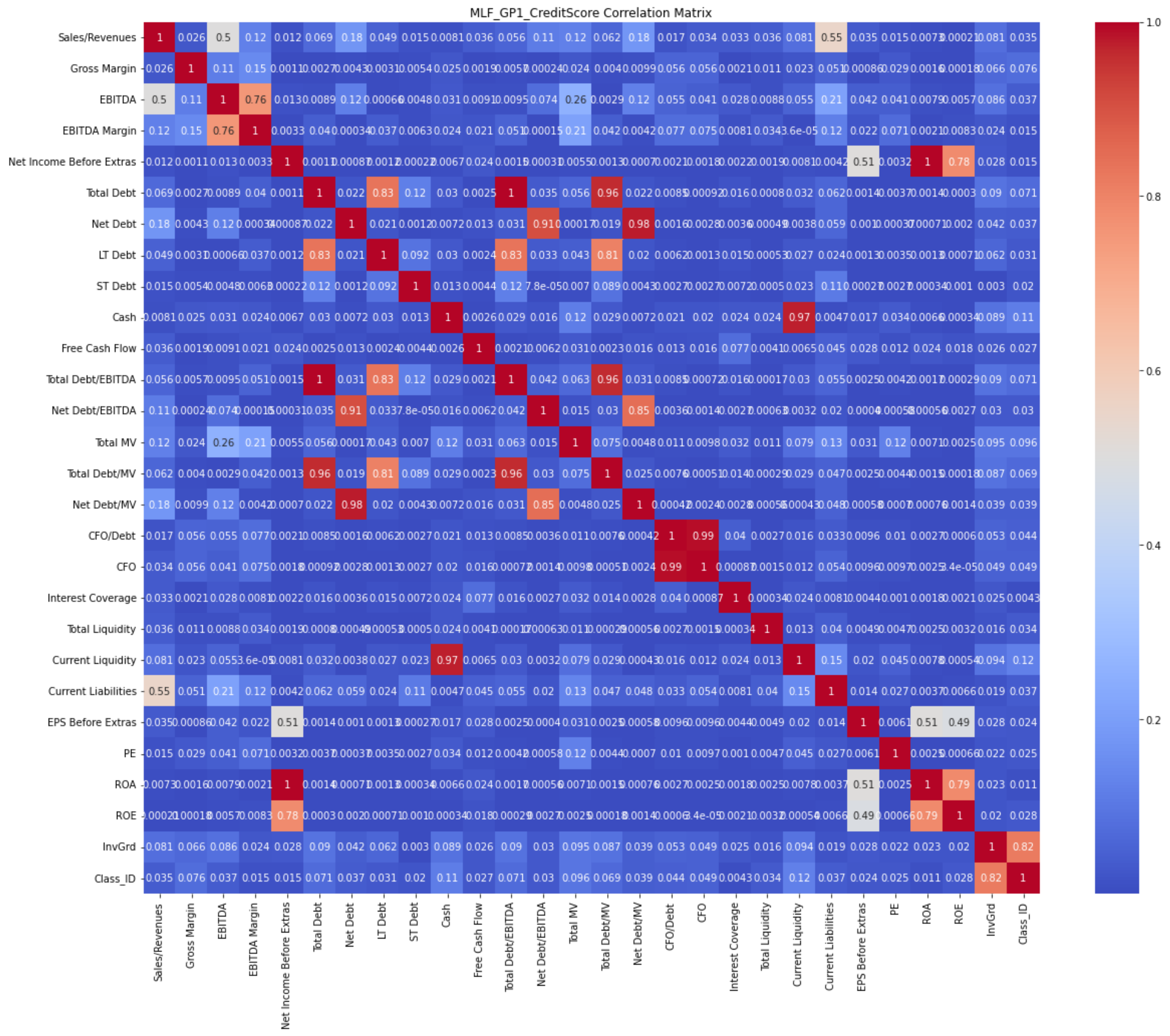


Figure 1.3: *Correlation matrix of the Moody's Credit Score Data Features*

1.2 Preprocessing, Feature Extraction, & Feature Selection

To prepare the dataset for supervised learning classification, the target variable of "Investment Grade" was used, which was already defined as a binary variable within the original CSV file. For the multiclass classification, the Moody's credit ratings were bucketed into four classes: A-Grade, which included grades Aaa, Aa1, Aa2, Aa3, A1, A2, and A3; Upper B-grade, which included Baa1, Baa2, and Baa3; Lower B-grade, which included Ba1, Ba2, Ba3, B1, B2, and B3; and C-Grade, which included Caa1, Caa2, Caa3, Ca, and C. By bucketing the ratings into their overarching letter assignment as opposed to each unique rating, this will reduce complexity in the model by reducing the number of target variables. To represent these four classes in the dataset, a "Class_ID" column was created, and class numbers are assigned according to Moody rating. If it received an A-grade, it was assigned 1, Upper B-grade was assigned 2, and so on until all data points were assigned their corresponding Class_ID. The data was then split using a stratified *train_test_split* with a 20% test size.

For both the multiclass and binary classifiers, due to the class imbalance between the number of investment grade versus non investment grade data in the binary case and between credit score ratings in the multiclass case, Synthetic Minority Oversampling Technique, or SMOTE, from the *imbalance* library was used to create an equal number of occurrences for both classes to prevent models from defaulting to the majority class. For the binary case, the original split of 1,030 investment grade bonds and 330 non-investment grade bonds was raised so that both classes contained 1,030 samples for each class. For the multiclass case, each grade letter was raised to contain 590 samples per class, matching the number of samples that were upper B-grade, the majority class in the dataset. The datasets for both the binary and multiclass models were then scaled using *StandardScaler()* to prepare for feature extraction.

Feature extraction was done for both the binary and multiclass classifiers. The goal was to address the correlation between certain features within the dataset, such as the almost perfect correlation between "Current Liquidity" and "Cash", the perfect correlation between "Total Debt" and "Total Debt/EBITDA" features, etc. For the binary classification, Principal Component Analysis (PCA) was performed, resulting in the following explained variances across principal components, as shown in Figure 1.4.

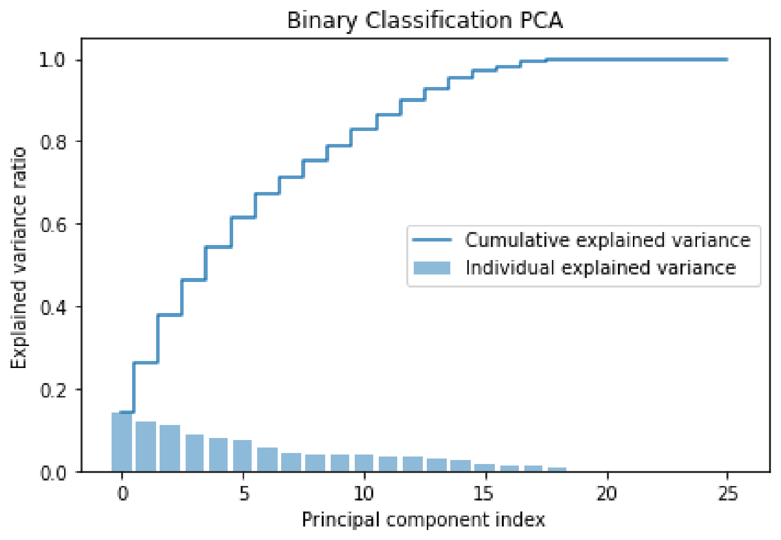


Figure 1.4: *Individual and cumulative explained variance for PCA components in Binary Classification*

As a result, the number of components that accounted for at least 95% of the cumulative explained variance was used in model fitting so that the dimensionality of the data could be reduced without losing a significant amount of the variability of the data itself. Going forward, 16 *n_components* were used when referring to the binary classification model done with PCA. Separate models are trained without the use of PCA to determine if performance improvement was seen in the data.

For the multi classification problem, Linear Discriminant Analysis (LDA) was used to help with class separability within the data. LDA resulted in three components that accounted for 65.82%, 26.02%, and 8.16% of the explained variance, respectively, as shown in Figure 1.5.

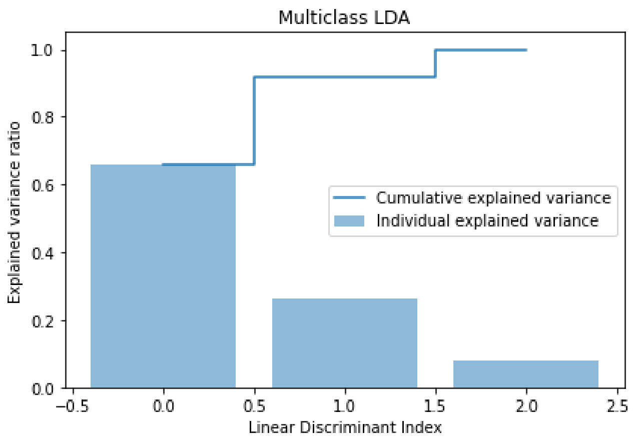


Figure 1.5: *Individual and cumulative explained variance for LDA discriminants in Multiclass Classification*

Similar to the binary classification, a separate set of models are trained without the use of LDA to determine how using this feature extraction technique affects performance improvement.

1.3 Model Fitting & Evaluation

For both the binary and multi classification problems, three types of classifiers both with and without feature extractions were considered: tree-based AdaBoost Classifier, Random Forest Classifier, and Support Vector Classification (SVC).

After hyperparameter tuning using GridSearchCV (see the following section *1.4 Hyperparameter Tuning* for more details), the models were fit using the training data using the hyperparameters generated as the best via GridSearchCV. Accuracy, precision, and recall metrics were used to evaluate model performance. Table 1.2 shows the resulting metrics for the binary classification.

Table 1.2: *Performance of the three classification models for binary classification, both with and without PCA. Best model test accuracies with and without PCA are shown in red.*

Models				Not Investment Grade		Investment Grade	
			Accuracy	Precision	Recall	Precision	Recall
1	**AdaBoost with PCA**	**Train**	**0.94**	**0.95**	**0.93**	**0.93**	**0.95**
		Test	**0.68**	**0.40**	**0.66**	**0.86**	**0.86**
2	**AdaBoost without PCA**	**Train**	**0.97**	**0.97**	**0.96**	**0.96**	**0.97**
		Test	**0.74**	**0.47**	**0.63**	**0.87**	**0.77**
3	**Random Forest with PCA**	**Train**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		Test	**0.78**	**0.54**	**0.60**	**0.87**	**0.84**
4	**Random Forest without PCA**	**Train**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
		Test	**0.86**	**0.69**	**0.81**	**0.93**	**0.88**

5	**SVC with PCA**	**Train**	**0.72**	**0.84**	**0.53**	**0.66**	**0.90**
		Test	**0.74**	**0.46**	**0.49**	**0.83**	**0.81**
6	**SVC without PCA**	**Train**	**0.72**	**0.84**	**0.53**	**0.66**	**0.90**
		Test	**0.74**	**0.46**	**0.49**	**0.83**	**0.81**

Model results were also illustrated with confusion matrices. The best model for binary classification was the random forest classifier done without PCA, with an accuracy of 0.86. AdaBoost mistakenly classified investment grade bonds as “not investment” grade more frequently than the other models. SVC also had trouble determining what was “not investment” grade as well. SVC did not have any performance changes with or without PCA either. Random Forest did the best, with the best metrics across precision, recall, and accuracy, and with the best positive classification rate for both classes, as shown in the classification matrix in Figure 1.6.

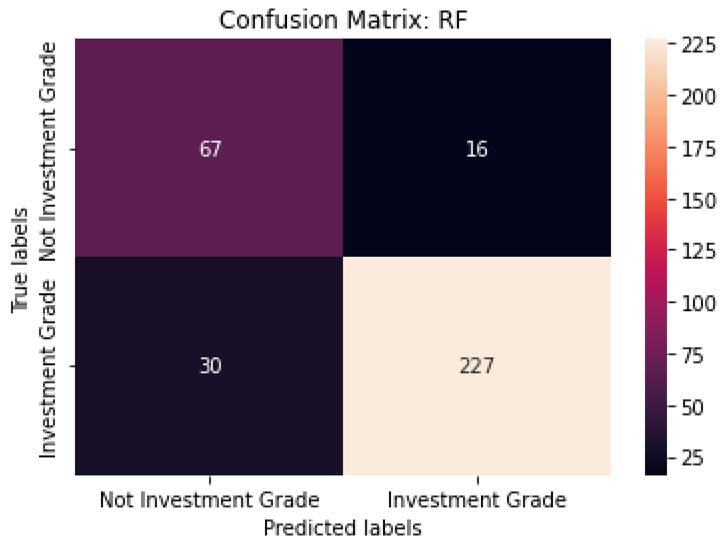


Figure 1.6: *Confusion matrix for the best binary classification model, the Random Forest Classifier without PCA*

After hyperparameter tuning, Table 1.3 shows the resulting metrics for the multiclass classification models.

Table 1.3: *Performance of the three classification models for multiclass classification, both with and without LDA. Best model test accuracies with and without PCA are shown in red.*

	Models			A-Grade		Upper B-Grade		Lower B-Grade		C-Grade	
			Accuracy	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
1	**AdaBoost with LDA**	**Train**	**0.49**	**0.38**	**0.22**	**0.37**	**0.48**	**0.34**	**0.38**	**0.86**	**0.88**
		Test	**0.34**	**0.33**	**0.17**	**0.46**	**0.44**	**0.26**	**0.38**	**0.00**	**0.00**
2	**AdaBoost without LDA**	**Train**	**0.60**	**0.47**	**0.70**	**0.49**	**0.19**	**0.49**	**0.58**	**0.99**	**0.91**
		Test	**0.42**	**0.42**	**0.66**	**0.55**	**0.20**	**0.38**	**0.52**	**0.00**	**0.00**
3	**Random Forest with LDA**	**Train**	**0.49**	**0.38**	**0.22**	**0.37**	**0.48**	**0.34**	**0.38**	**0.86**	**0.88**
		Test	**0.47**	**0.52**	**0.55**	**0.53**	**0.42**	**0.42**	**0.47**	**0.00**	**0.00**
4	**Random Forest without LDA**	**Train**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		Test	**0.77**	**0.75**	**0.78**	**0.80**	**0.76**	**0.76**	**0.80**	**0.00**	**0.00**
5	**SVC with LDA**	**Train**	**0.61**	**0.47**	**0.75**	**0.62**	**0.33**	**0.67**	**0.43**	**0.75**	**0.91**
		Test	**0.43**	**0.44**	**0.65**	**0.60**	**0.28**	**0.50**	**0.41**	**0.02**	**0.50**
6	**SVC without LDA**	**Train**	**0.67**	**0.48**	**0.87**	**0.67**	**0.35**	**0.79**	**0.46**	**0.91**	**1.00**
		Test	**0.49**	**0.47**	**0.85**	**0.60**	**0.26**	**0.55**	**0.44**	**0.00**	**0.00**

Confusion matrices were also generated for each of the models. The best model overall was the Random Forest Classifier done without LDA by a long shot across all metrics. None of the models were able to correctly classify C-Grade bonds. This could be due to the very limited amount of unique data points in the original dataset, as there were only 7 C-grade bonds out of 1,700 observations, which was an issue SMOTE could not overcome. All of the other models, with and without LDA, had high misclassification of bonds across A-grade, upper B-Grade, and lower B-grades, as if the models could not define any class separability. Upper B-grade bonds were the most misclassified as lower B-grade bonds, which means the models are not useful for discriminating between what bonds should be included in investment portfolios in the future. Random Forest without LDA was able to better distinguish between these two. While A-grade bonds in this model were sometimes classified as Upper B-grade bonda and vice versa, this an acceptable error for the model since they both are overall considered acceptable investment grades. The confusion for the Random Forest model without LDA for the multiclass classification is shown below in Figure 1.7.

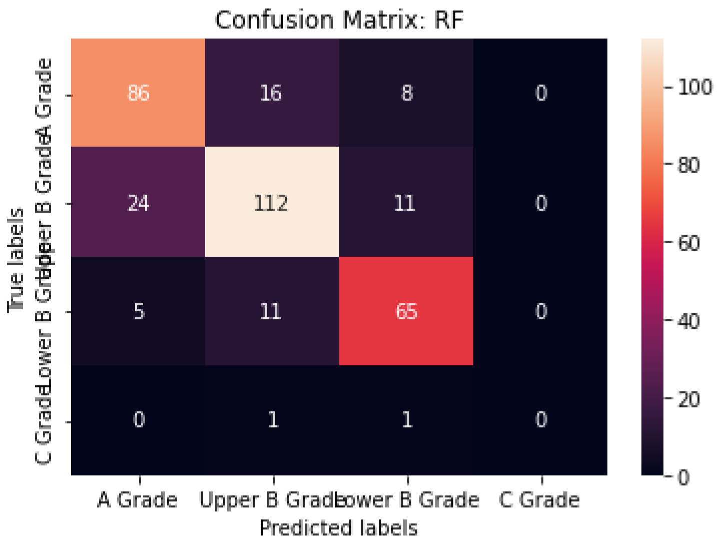


Figure 1.7: *Confusion matrix for the best multiclass classification model, the Random Forest Classifier without LDA*

1.4 Hyperparameter Tuning

GridSearchCV was used for hyperparameter tuning for all models for both the binary and multiclass classification models. AdaBoost and Random Forest models were tuned to optimize *n_estimators*, with GridSearchCV iterating over values of 100, 200, 300, 400, 500, 750, 1000, 1500, and 2000 as potential hyperparameter values. SVC was tuned to optimize both *C* and *kernel* hyperparameters, as well as the *degree* for the 'poly' kernel option. The values of *C* in the parameter grid included 0.1, 0.2, 0.5, 1.0, 1.5, and 2.0; the values of *kernel* included linear, poly, and rbf; and the values of *degree* included 2, 3, and 4.

Table 1.4 shows the resulting hyperparameters that were determined to be the best parameters based on the GridSearchCV results for the binary and multiclass classification models.

***Table 1.4**: Best hyperparameters resulting from GridSearchCV for the models fit, including both binary and multiclass classifiers*

	Models		Hyperparameters			
			n_estimators	**C**	**Kernel**	**Degree (if Kernel is *poly*)**
1	**AdaBoost with feature extraction**	**Binary**	**1500**	**-**	**-**	**-**
		Multiclass	**100**	**-**	**-**	**-**
2	**AdaBoost without feature extraction**	**Binary**	**750**	**-**	**-**	**-**
		Multiclass	**2000**	**-**	**-**	**-**
3	**Random Forest with feature extraction**	**Binary**	**750**	**-**	**-**	**-**
		Multiclass	**750**	**-**	**-**	**-**
4	**Random Forest without feature extraction**	**Binary**	**400**	**-**	**-**	**-**
		Multiclass	**300**	**-**	**-**	**-**
5	**SVC with feature extraction**	**Binary**	**-**	**2.0**	**RBF**	**-**
		Multiclass	**-**	**2.0**	**RBF**	**-**
6	**SVC without feature extraction**	**Binary**	**-**	**2.0**	**RBF**	**-**
		Multiclass	**-**	**2.0**	**RBF**	**-**

Interesting trends in hyperparameters was the consistent choice of hyperparameters in SVC despite changing the number of classes or if feature extraction was used or not. Additionally, the number of estimators in the random forest both decreased when LDA was not used for both the binary and multiclass classifiers. AdaBoost, on the other hand, exhibited the same trend for its binary classifier, but the number of estimators actually increased in the multiclass classifier once LDA was removed.

1.5 Ensembling

Ensembling was used in two of the models trained to solve this problem. AdaBoost was used, which is a boosting ensembling method using a collection of weak learners; in this case, decision tree stumps were used to be the weak learners. The other ensembling method used was Random Forest, which ensembles Decision Trees as well, but the trees are not necessarily stumps like what was used for AdaBoost. These two methods, compared to SVC, were noticeably less prone to overfitting. Since first attempts did not use SMOTE to address class imbalance, SVC would default to classifying to the majority class, where AdaBoost and Random Forest were able to perform better despite the class imbalance. Even after SMOTE was used, SVC still would default to one of the classes for classification. SVC continued to exhibit low performance even after SMOTE was used, so utilizing ensembling in the classification problems for both the binary and multiclass classifiers was worthwhile.

1.6 Conclusions

This chapter we apply the credit rating database to do a series of classifications to successfully classify investment bonds as either investment grade or not. The exploratory data analysis shows that most credits are rated as “Baa2” according to Moody’s rating score evaluation. The heatmap of correlation coefficients among all variables indicates that it is needed to reduce the dimension of data by feature extraction and/or selection. As a result, PCA and LDA were attempted for feature extraction. Two avenues of data classification were done: binary classification by separating what is or is not investment grade, and multiclass classification by classifying Moody’s scores as A-grade, Upper B-grade, Lower B-grade, and C-grade. Despite using feature extraction techniques, the best model in both avenues was the Random Forest Classifier done without feature extraction. Next steps to improve performance would be to attempt other model types or feature selection techniques so that dimensionality reduction is useful towards model fitting. Additionally, spending more time on how to better separate upper and lower B-grade bonds so that the model can be used to accurately predict new bonds in practice, which would allow the model to be integrated to improve investor workflow.

CHAPTER 2: Economic Cycle Regression

2.1 Introduction & Exploratory Data Analysis (EDA)

The Second data is derived from a published paper "*Why does the Paper-Bill spread predict real economic activity?*" written by Friedman and Kuttner in 1993. This paper illustrates the spread on commercial paper, a short term form of corporate borrowing, and the US Treasury bill widens before recessions and contracts after and could be a useful candidate for real economic activity. Although the raw data is formatted with time-series, for the purpose of our group project, we ignore the characteristic of time since this would not substantially affect our results. We apply the PCA and regression techniques, to employ a series of models to predict the percent change in the USHPCI 3, 6 and 9 months ahead (named in the dataset as "PCT 3MO FWD", "PCT 6MO FWD" and "PCT 9MO FWD", respectively) .

First, we conduct a statistical descriptive analysis of our data. The results are displayed in Table 2.1. Due to the space constraint, we do not completely show the statistical results for all variables, while selected variables and the three target variables of interest (3MO, 6MO, and 9MO) are reported in Table 2.1. One can find that the yields of different treasury bills are not substantially different as slight differences could be observed.

Table 2.1: *Descriptive statistics for selected variables in the Economic Cycle Dataset*

	T1Y	T2Y	T3Y	T5Y	T7Y	T10Y	C_T1Y	C_T1Y	C_T1Y	PCT 3MO FWD	PCT 6MO FWD	PCT 9MO FWD
count	223.00	223.00	223.00	223.00	223.00	223.00	223.00	223.00	223.00	223.00	223.00	223.00
mean	8.03	8.41	8.56	8.81	8.98	9.07	0.98	0.98	0.98	0.01	0.01	0.02
std	3.16	2.95	2.82	2.65	2.54	2.45	0.09	0.08	0.07	0.00	0.01	0.01
min	3.18	3.84	4.17	4.71	5.05	5.33	0.72	0.71	0.70	-0.01	-0.01	-0.01
25%	5.73	6.18	6.41	6.70	6.97	7.18	0.93	0.94	0.95	0.01	0.01	0.01
50%	7.67	8.00	8.13	8.33	8.52	8.61	0.97	0.98	0.98	0.01	0.02	0.02
75%	9.84	10.08	10.38	10.53	10.64	10.69	1.03	1.03	1.02	0.01	0.02	0.03
max	16.72	16.46	16.22	15.93	15.65	15.32	1.34	1.28	1.22	0.02	0.04	0.05

Second, considering the distributions of the Treasury bill yields further, we plot the distributions of treasury bill yields indexes with different maturities with boxplot (named "T#Y Index" in the dataset for maturity replacing #). The results are shown in Figure 2.1. One could find that no substantial differences can be observed from the inspection of the figure. One can see the median value of each yield slightly increases with the maturity going far.

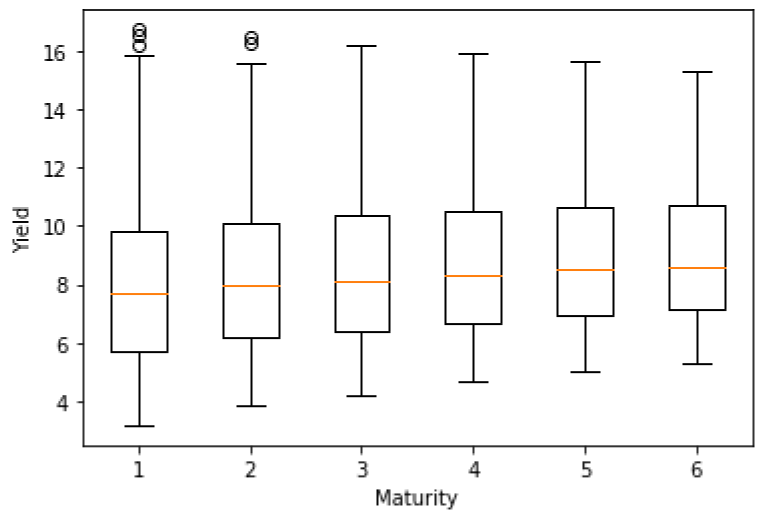


Figure 2.1: *The distributions of Treasury bill yields with different maturities*

Similarly, the distributions of commercial paper yields with different maturities are shown in Figure 2.2 (named "CP#M" in the dataset with each maturity replacing #). One can arrive at a similar conclusion as the one drawn for Figure 2.1, where there are little differences between these variables. The yields of commercial papers with different maturities are generally smaller than the yields for treasury bills.

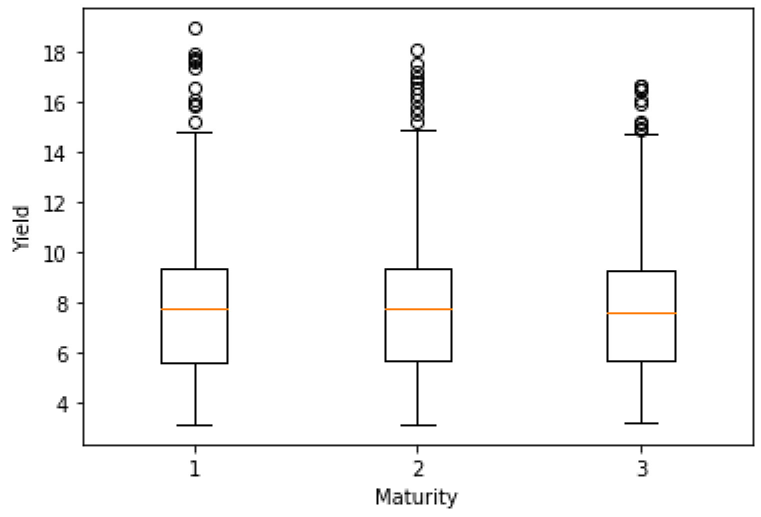


Figure 2.2: *The distributions of commercial paper yields with different maturities*

Finally, we analyze the correlations of Treasury bill yields with different maturities by heatmap and the results are shown in Figure 2.3. The correlations of Treasury bills with different maturities are generally high, most of which are above 0.95, with greater correlation between those maturities that are closer in age. This confirms that the yields of Treasury bills with different maturities are generally similar.

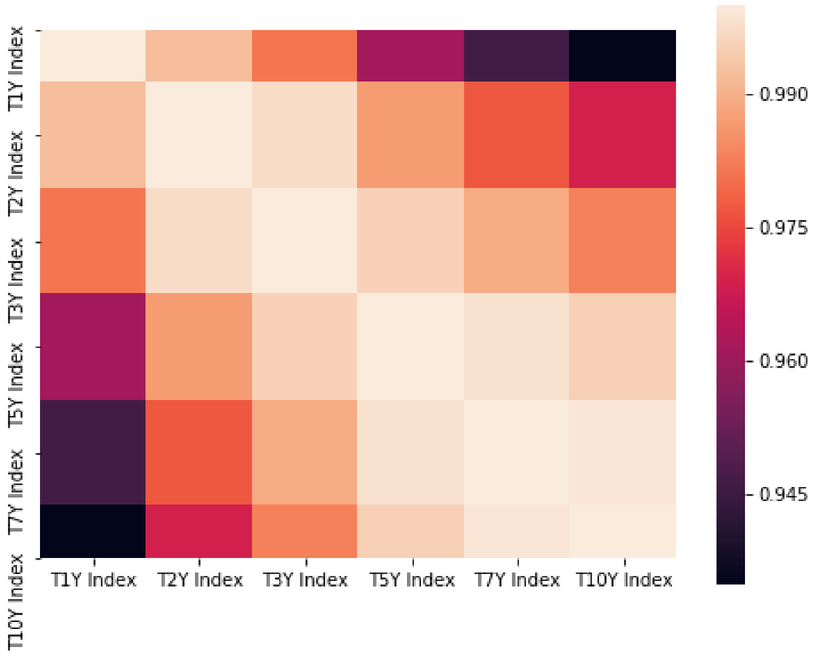


Figure 2.3: *The correlations of Treasury bill yields with different maturities.*

With similar logic, we plot a correlation heatmap for the commercial paper yields with different maturities, and the results are shown in Figure 2.4. These yields are also highly correlated across maturities, with greater correlation for yields closer to each other in maturity.

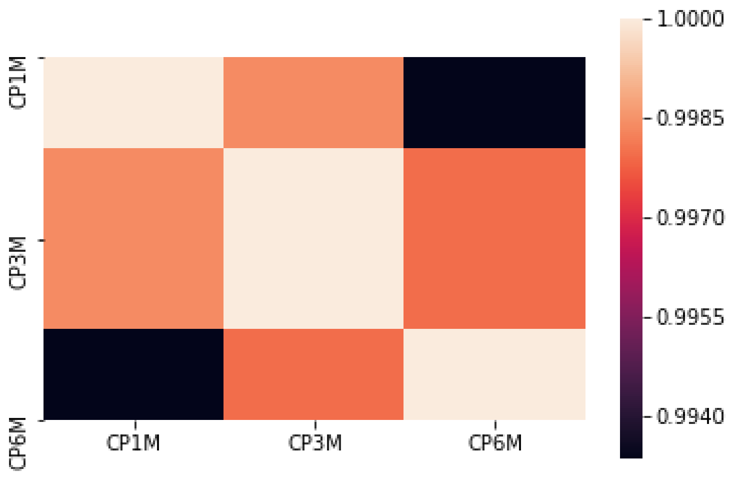


Figure 2.4: *The correlations of commercial paper yields with different maturities.*

2.2 Preprocessing, Feature Extraction, & Feature Selection

To preprocess the data, the "USHPCI" and date columns were dropped. Because the data is time dependent and USHPCI would be directly correlated with the percent change in USHPCI, this forces the model to draw correlations between the other features in the dataset to the target variable.

We standardized all the features using *StandardScaler()* and used Principal Component Analysis (PCA) to compress the first six features (T1Y Index, T2Y Index, T3Y Index, T5Y Index, T7Y Index, T10Y Index) into one principal component since these features are highly positively correlated with each other, as seen from the heatmap in the previous section (Figure 2.3). The individual and cumulative explained variance ratios from PCA are shown in Figure 2.5.

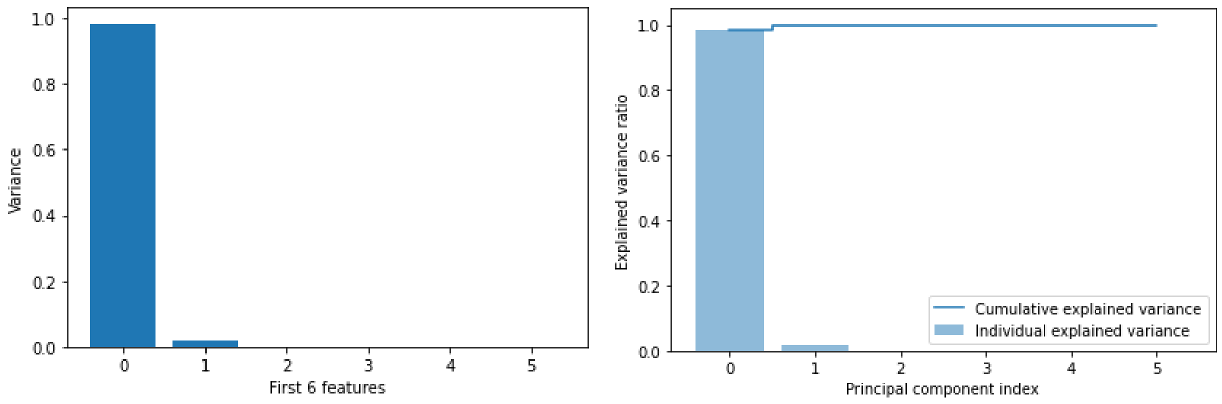
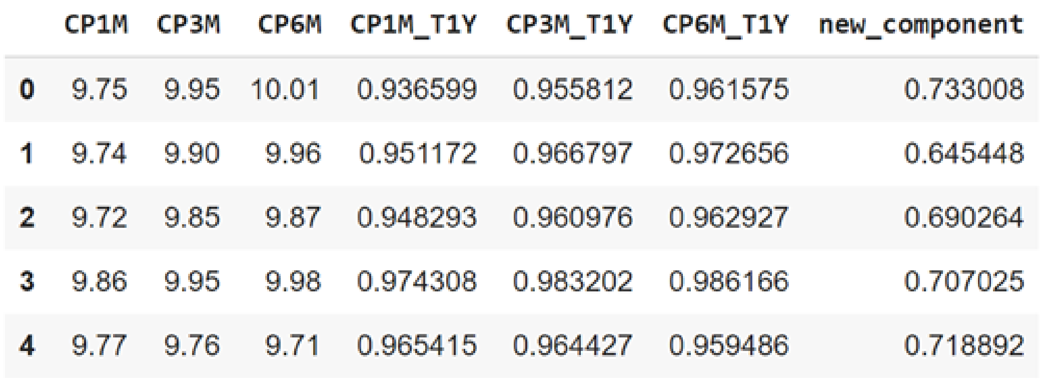


Figure 2.5: *Principal component analysis (PCA) on the Economic Cycle Data using the first 6 features*

The features we used to predict percent change in the Index 3-, 6-, and 9-month ahead (targets 3MO, 6MO, and 9MO, respectively, hereafter) included commercial paper yields at 1, 3, 6 month maturities; the ratio of commercial paper yields to the 1 year (1Y) Treasury yield for 1, 3, and 6 month maturities; and the new component generated from PCA based on the treasury yield indexes, which was appended to the original dataframe, as shown in the Figure 2.6.



	CP1M	CP3M	CP6M	CP1M_T1Y	CP3M_T1Y	CP6M_T1Y	new_component
0	9.75	9.95	10.01	0.936599	0.955812	0.961575	0.733008
1	9.74	9.90	9.96	0.951172	0.966797	0.972656	0.645448
2	9.72	9.85	9.87	0.948293	0.960976	0.962927	0.690264
3	9.86	9.95	9.98	0.974308	0.983202	0.986166	0.707025
4	9.77	9.76	9.71	0.965415	0.964427	0.959486	0.718892

Figure 2.6. *New data frame created by appending the PCA components*

Based on the new dataframe, we draw a new heatmap to see the correlation between each feature as shown in Figure 2.7. The Pearson's correlation coefficient is between 0.055 and 1, where 1 is perfect correlation and 0.055 is almost uncorrelated. While the features that only differ by maturity are still highly correlated with each other, the features in general are not overly correlated to each other and are ready to be modeled.

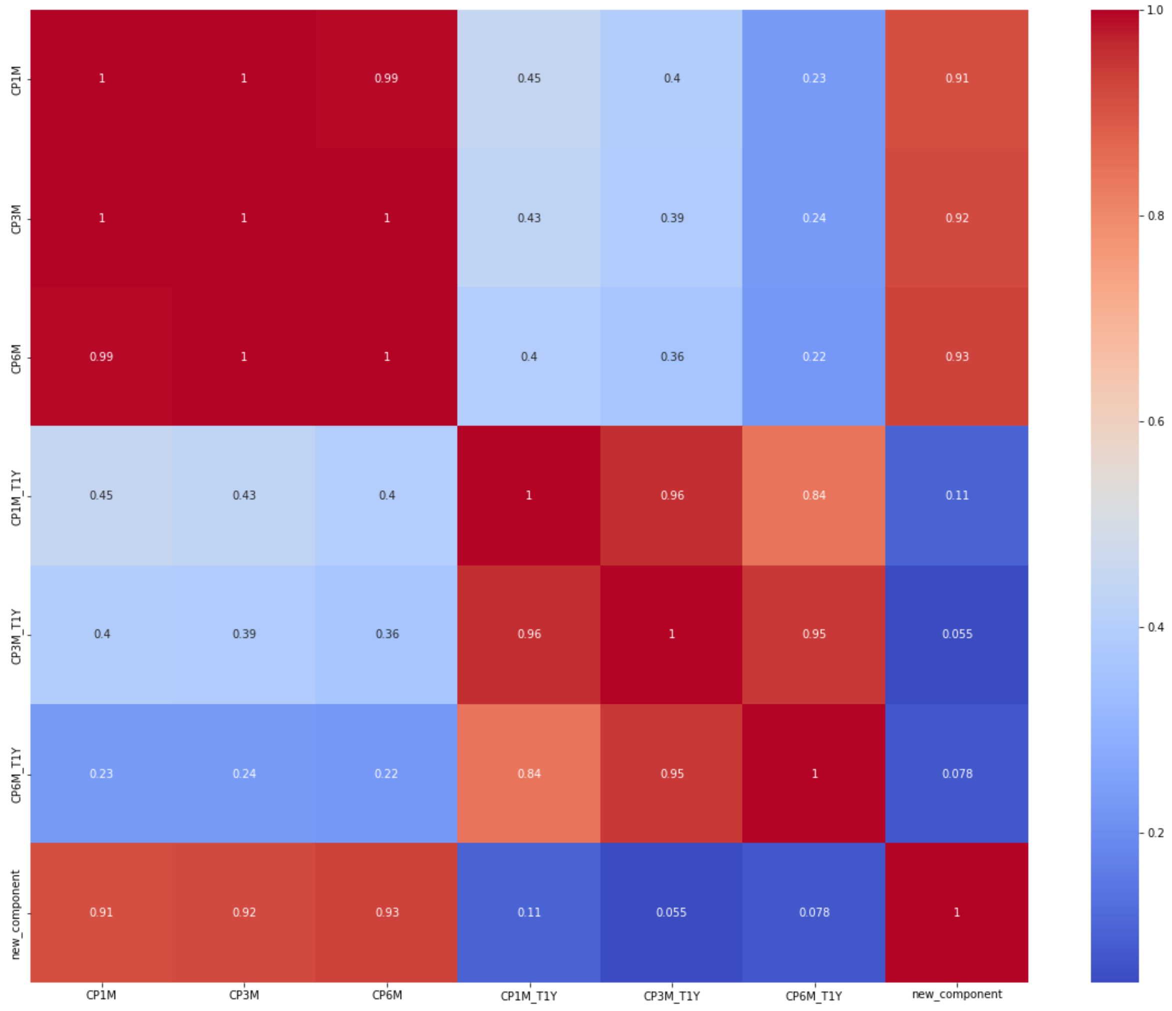


***Figure 2.7**: Correlation Matrix of the features in the dataset*

We predicted the percentage change of 3MO, 6MO, and 9MO targets, separately, by different regression models and estimated their generalization performance via statistical index mean squared error (MSE) and coefficient of determination (R^2). The models applied in the regression task included linear regression, polynomial regression, Ridge regression, Elastic Net, Support Vector Regression (SVR), Decision Tree Regressor, Random Forest Regressor, AdaBoostRegressor, and Multi-Layer Perceptron (MLP) Regressor. R^2 value can be positive, 0, or negative, where positive is a perfect fit, 0 is uncorrelated, and negative value indicates an inverse fit.

2.3 Model Fitting, Ensembling & Evaluation

Table 2.2 shows MSE and R^2 values of each model. The tree-based models (Decision Tree Regressor, Random Forest Regressor, and AdaBoost Regressor) have the best performance, compared to the other models that suffer from overfitting, as they perform well on training data but show a negative R^2 value on the on the test set. Moreover, the prediction of the 9MO target has the best performance in each model in contrast to 3MO and 6MO targets. We inferred that financial data fluctuates a lot in a short period of time and tends to appear more stable after a long period of time. Take a K line in a stock chart for example. When zooming in the chart, you will see the K line is very jagged, with sharp increases and decreases in the trend line. But once zoomed out, the line appears to be more stable with a more visible increasing or decreasing trend line. Therefore, the discussion in the following section will mainly focus on the 9MO target.

***Table 2.2**: Performance of nine regression models as shown by MSE and R^2 metrics*

	Models		**y-targets**					
			3MO		**6MO**		**9MO**	
			MSE	**R^2**	**MSE**	**R^2**	**MSE**	**R^2**
1	**Linear regression**	**Train**	**0.000239**	**-10.137**	**0.000108**	**-0.435**	**0.000109**	**0.334**
		Test	**0.000274**	**-9.121**	**0.000161**	**-0.583**	**0.000153**	**0.159**
2	**Polynomial regression**	**Train**	**0.000276**	**-11.837**	**0.000076**	**-0.018**	**0.000014**	**0.916**
		Test	**5.891539**	**-217313**	**5.886279**	**-57700**	**5.880181**	**-32374**
3	**Ridge regression**	**Train**	**0.000016**	**0.256**	**0.000054**	**0.280**	**0.000111**	**-1.047**
		Test	**0.000026**	**0.044**	**0.000082**	**0.198**	**0.000135**	**0.256**
4	**Elastic Net**	**Train**	**0.000022**	**0.000**	**0.000075**	**0.000**	**0.000164**	**0.000**
		Test	**0.000028**	**-0.022**	**0.000102**	**0.000**	**0.000182**	**-0.002**

5	**SVR**	**Train**	**0.000026**	**-0.218**	**0.000078**	**-0.044**	**0.000177**	**-0.080**
		Test	**0.000036**	**-0.318**	**0.000106**	**-0.038**	**0.000199**	**-0.097**
6	**Decision Tree Regressor**	**Train**	**0.000010**	**0.531**	**0.000035**	**0.537**	**0.000054**	**0.670**
		Test	**0.000021**	**0.232**	**0.000076**	**0.259**	**0.000096**	**0.471**
7	**Random Forest Regressor**	**Train**	**0.000009**	**0.560**	**0.000028**	**0.623**	**0.000050**	**0.697**
		Test	**0.000019**	**0.313**	**0.000056**	**0.447**	**0.000085**	**0.530**
8	**AdaBoost Regressor**	**Train**	**0.000009**	**0.564**	**0.000027**	**0.643**	**0.000048**	**0.710**
		Test	**0.000018**	**0.329**	**0.000051**	**0.501**	**0.000077**	**0.576**
9	**Multilayer Perceptron Regressor**	**Train**	**0.041050**	**-1907**	**0.040994**	**-545**	**0.040990**	**-249**
		Test	**0.038040**	**-1402**	**0.037830**	**-369**	**0.037667**	**-206**

2.4 Hyperparameter Tuning

To find the optimal combination of hyperparameter values for each model, we applied GridSearchCV (with 5-folds). The average CV scores of each model are shown in Table 2.3.

Table 2.3: *The mean and standard deviation of 5-fold CV scores for each model.*

Models	**CV scores (k=5)**						
	y-targets	**PCT 3MO FWD**		**PCT 6MO FWD**		**PCT 9MO FWD**	
		Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
Ridge regression	**Train**	**0.136**	**0.143**	**0.193**	**0.111**	**0.261**	**0.098**
	Test	**0.075**	**0.357**	**0.252**	**0.233**	**0.309**	**0.230**

Elastic Net	**Train**	**0.096**	**0.107**	**0.177**	**0.083**	**0.247**	**0.074**
	Test	**0.056**	**0.303**	**0.216**	**0.290**	**0.255**	**0.256**
SVR	**Train**	**-0.319**	**0.217**	**-0.084**	**0.065**	**-0.108**	**0.084**
	Test	**-0.070**	**0.054**	**-0.100**	**0.081**	**-0.157**	**0.100**
Decision Tree Regressor	**Train**	**0.190**	**0.130**	**0.264**	**0.156**	**0.306**	**0.162**
	Test	**0.313**	**0.217**	**0.535**	**0.144**	**0.450**	**0.117**
Random Forest Regressor	**Train**	**0.174**	**0.170**	**0.370**	**0.251**	**0.539**	**0.152**
	Test	**0.307**	**0.148**	**0.502**	**0.169**	**0.488**	**0.153**
AdaBoost Regressor	**Train**	**0.032**	**0.187**	**0.216**	**0.201**	**0.423**	**0.162**
	Test	**0.090**	**0.435**	**0.444**	**0.162**	**0.298**	**0.186**
Multilayer Perceptron Regressor	**Train**	**-213.955**	**168.206**	**-58.600**	**38.999**	**-19.659**	**7.232**
	Test	**-215.247**	**76.566**	**-38.655**	**13.831**	**-23.752**	**8.106**

The Decision Tree model was tuned to optimize *max_depth*, *max_features*, and *criterion*. Random Forest model was tuned to optimize both *max_depth* and *max_features* as well as *n_estimators* and *min_samples_split*. AdaBoost was tuned to optimize *n_estimators* and *loss function*. The value of each parameter in the parameter grid can be found in greater detail in the code, provided through a GitHub link on the last page (see APPENDIX).

After tuning hyperparameters, the performances of the given models were greatly improved, as shown in Table 2.4.

Table 2.4: *The mean and standard deviation of 5-fold CV scores for each model.*

Models		y-targets					
		PCT 3MO FWD		PCT 6MO FWD		PCT 9MO FWD	
		MSE	R^2	MSE	R^2	MSE	R^2
Ridge regression	**Train**	**0.000015**	**0.323**	**0.000052**	**0.309**	**0.000109**	**-1.04**
	Test	**0.000032**	**-0.19**	**0.000096**	**0.062**	**0.00014**	**0.210**
Elastic Net	**Train**	**0.00002**	**0.243**	**0.00005**	**0.277**	**0.00011**	**0.330**
	Test	**0.00003**	**0.060**	**0.00008**	**0.202**	**0.000140**	**0.230**
SVR	**Train**	**0.000026**	**-0.22**	**0.000078**	**-0.04**	**0.000177**	**-0.08**
	Test	**0.000036**	**-0.32**	**0.000106**	**-0.04**	**0.000199**	**-0.10**
Decision Tree Regressor	**Train**	**0.000016**	**0.277**	**0.000049**	**0.341**	**0.000071**	**0.568**
	Test	**0.000024**	**0.111**	**0.000058**	**0.432**	**0.000083**	**0.540**
Random Forest Regressor	**Train**	**0.00001**	**0.673**	**0.00001**	**0.860**	**0.00001**	**0.941**
	Test	**0.00002**	**0.447**	**0.00005**	**0.542**	**0.00007**	**0.642**
AdaBoost Regressor	**Train**	**0.00000**	**0.994**	**0.00000**	**0.995**	**0.00000**	**0.995**
	Test	**0.00001**	**0.471**	**0.00005**	**0.518**	**0.00006**	**0.655**
Multilayer Perceptron Regressor	**Train**	**0.00187**	**-85.9**	**0.00207**	**-26.6**	**0.00267**	**-15.3**
	Test	**0.00309**	**-113**	**0.00329**	**-31.2**	**0.00344**	**-18.0**

The R^2 of the test set from Decision Tree Regressor was increased to 0.540 from 0.471. Although the R^2 value of the training set decreased (from 0.670 to 0.568), the overall performance has improved since the overfitting problem was solved (R^2 value of the training and test sets are closer). For Random Forest and AdaBoost models, the R^2 values of the training set were increased to 0.941 and 0.995, respectively. However, for the test set, their R^2 values were only 0.642 and 0.655, respectively. It is obvious that overfitting occurs after GridSearchCV. Therefore, in the regression task, the Decision Tree model has the best performance.

2.5 Conclusions

Principal Component Analysis (PCA) compresses the first six features into one principal component while maintaining important information in the dataset. The Decision Tree model has the best performance for the regression task with R^2 values of 0.568 and 0.540 for the training set and test set, respectively. Thus, the models support the paper where paper bill spread was claimed to predict real economic activity, as shown in the percent change in USPHCI. The models also exhibited best performance when predicting the percent change in USPHCI over 9 months compared to 6 months and 3 months out, which can be explained by the financial data appearing more stable in the long-term than in the short-term. Future considerations for prediction would be to try and integrate the time aspect into the data to further improve performance, such as using a “rolling window” in which recent data is weighted more than historic data, which allows for more accurate predictions. Additionally, considering different feature engineering approaches to further reduce dimensionality without losing performance would be a means to improve overall performance across target variables.

3. APPENDIX

GitHub Repository

The project repository for the code described in this report can be accessed here. The EDA for the regression and classification problem are separated into two files, and the model fitting and evaluation for the classification and regression tasks are in their own respective files as well: https://github.com/eemayes2/IE517MLF_Group_project