Credit Card Fraud Detection

Machine Learning Engineer Capstone Project

By Patrick Wellins

Project Overview

Over the years, billions of dollars have been lost to fraudulent credit card charges. In an article by the Economist (2014), it is shown that in 2012 the total amount of fraudulent credit card charges worldwide amounted to over 10 billion dollars. Technology to detect fraudulent charges is currently employed, and many startup technology companies are using machine learning for fraud prevention (CBINSIGHTS 2017). One area of machine learning called supervised learning involves learning patterns that exist in labeled datasets. A model that learns patterns in labeled data can make predictions on unlabeled data; these predictions could be a class that a variable belongs to or real-valued numbers.

This report will describe the process of building a supervised learning model based on a Kaggle dataset (Dal Pozzolo et al. 2015) that contains 284,807 credit card charges that are labeled as genuine or fraudulent. The charges occurred over a two-day period in 2013 and were made by European cardholders. There is a total of 31 columns in the dataset that represent different variables. The variable 'Class' is the target variable that is to be predicted by the supervised learning model. The values that the variable 'Class' take on are either 0 or 1. Class 1 represents fraudulent charges, and Class 0 represents genuine charges. Out of the 284,807 charges, 492 (0.17%) are fraudulent. In this report, a step by step process will be demonstrated to show the development of a supervised learning model that makes predictions that a given charge is genuine or fraudulent.

The objective is to build a classification model that achieves a F1 score of >= 0.90 on the test set of credit card charges and exceeds all set benchmark scores. For this to be accomplished, the model will have to learn key differences between the independent variables of genuine and fraudulent charges. Once these differences have been learned, the model will make predictions on the test set. It is important to note that the test set consists of data points that the model has not been trained on. The goal is that the model generalizes well and makes class predictions on the test set that match the actual class label. Before model development, the data will be explored statistically and visually. Some statistical and visual techniques that will be employed include tests for normality of data, outlier detection, and visualizing the patterns of the two classes in scatter plots. Multiple models will be trained, and the strongest performing model will be optimized. The performance of the optimized model will be compared to benchmark performance metrics which include the accuracy of a naïve prediction model and an array of metrics that a logistic regression model attains on the test set. The F1 score of the final model will provide concrete evidence of model performance. A conclusion will be made about the quality of the model that is based on prediction performance on the test set.

The programming language used to solve this problem is Python. The libraries used for this project are pandas, random, numpy, sklearn, seaborn, matplotlib.pyplot,IPython.display, scipy, statistics, warnings, mpl_toolkits.mplot3d.

Problem Statement

The problem to be solved is classifying fraudulent and genuine credit card transactions in the test set with a high-level of performance defined by a F1 score of >= 0.9. To achieve the goal of a F1score >= 0.9 several steps will need to be carried out. The first step is to understand the data statistically and visually. Statistical analysis of the data will be informative regarding how the data should be preprocessed. Twenty-eight of the feature variables in this dataset are already standardized. Standardized variables allow for model parameters to be updated uniformly while the gradient descent algorithm is being carried out. It is possible that only a subset of the 30 feature variables will be predictive. To select the most predictive variables, the feature importance method that corresponds to a random forest classifier will be employed to choose the most predictive feature variables. Reducing the number of feature variables will allow for faster training time. Additionally, redundant or non-predictive variables do not contribute to higher model performance. After obtaining the most predictive feature variables, logistic regression, support vector machine, and decision tree models will be trained. These trained models will then make predictions on the test set of charges, and four metrics will be recorded. These metrics include accuracy, precision, recall and F1 score. The model with the highest F1 score will be further refined with hyper-parameter optimization. To choose the best hyper-parameters for the selected model, k-fold cross-validation and grid search will be employed. The process of cross-validation and grid search will test which combination of hyper-parameters have the highest validation score. This set of hyper-parameters will be chosen for the model, and the model will make predictions on the test set. The test set performance metrics will be provided and will be compared to the objective of a F1 score >= 0.90 and benchmarks.

Metrics

Four metrics will be used to evaluate the performance of all models built for classifying credit card charges. These four metrics include accuracy, precision, recall, and F1. A description of each of these metrics is provided along with how each metric integrates into the problem of classifying charges. The preeminent metric is the F1 score; this metric is the most effective for conveying model performance results due to the imbalance of the data set.

Accuracy is defined by #True Positives + #True Negatives / #False Positives + # False Negatives + #True Positives + #True Negatives. Accuracy is the percentage of predictions that match the actual class label. In the context of this problem, accuracy is not the best metric to evaluate model performance. Predicting every transaction as genuine will return an accuracy score of ~ 0.99827, this level of accuracy will be used as a benchmark for model performance. The high score of the naïve prediction model displays the ease of generating a high accuracy score when a dataset is highly imbalanced.

Precision is defined by #True Positives / #True Positives + #False Positives. Precision measures how well the model avoids making false positive predictions. A score of 1.0 indicates that of all positive predictions made there are no false positives generated. In the context of this problem,

precision is important for the following reason; a false positive has the potential to stop a person from making a legitimate purchase.

Recall is defined by #True Positives / #True Positives + #False Negatives. The denominator of this metric is equivalent to the total true positives in the dataset. A recall rate of 1.0 indicates that the model correctly identified every true positive in the dataset. The higher the recall is, the more fraudulent charges that have been correctly identified.

F1 is defined by 2 x ((Precision x Recall) / (Precision + Recall)). The F1 score is the harmonic mean of Precision and Recall. The F1 score is similar to the average of precision and recall. The F1 score will be the preeminent metric for evaluating model performance, given that F1 measures a combination precision and recall. The F1 score is a well-rounded evaluation metric for model performance.

Data Exploration

The dataset contains transactions made by European cardholders over a two-day period during September 2013. The dataset contains information for 284,807 charges, of the 284,807 charges 492 are fraudulent. The dataset is highly unbalanced; class 1 accounts for 0.172% of all transactions.

There is a total of 30 feature variables, the 28 variables beginning with 'V' are 28 principal components of the original data. The labels for the original 28 features in the data have been removed for confidentiality reasons. In addition to the 28 principal component variables, there are the variables 'Time' and 'Amount.' The unit of the 'Time' variable is in seconds. The currency unit is not specified for 'Amount.'

The variable ‘Amount’ corresponds to the amount of the transaction. The average of 'Amount' is approximately 88.35. The 'Amount' variable is not normally distributed, the mean value of 88.35 is distant from the median value of 22. Figure 4. is a QQ-plot that illustrates that 'Amount' is not normally distributed; if the values of 'Amount' were normally distributed the blue data points would track the red line in the QQ-plot. The 'Amount' variable has been tested for normality with the Shapiro-Wilk test; the corresponding p-value is 0. When testing for normality with the Shapiro-Wilk test, generally an alpha level of 0.05 is selected, and if the p-value is less than the alpha value there is a failure to accept the null hypothesis of normality. In this case, the p-value is 0, indicating a failure to accept that the 'Amount' variable is normally distributed.

The transactions took place over a 2-day period or over 172,792 seconds. Dividing 172,792 seconds by 3,600 seconds returns 48 hours. To get a better understanding of time, the seconds were converted to hours and the hours have been modularized so after the 24th hour the hours go to 0, this can be seen in figures 7 - 9. The hour 16 had the most transactions over the two-day period. Approximately 67 percent of the charges occurred during the second halves of each day.

The variables 'V1', 'V2', ... , 'V28' are principal components of the original feature variables. It is unfortunate that there are no original names for these features. However, insights can still be

gained from these unlabeled variables regarding the difference between fraudulent and genuine charges. The variables 'V1', 'V2', ... , 'V28' and not normally distributed and contain many outliers. Non-normality of these variables is shown by the Shapiro-Wilk test, p-values for the Shapiro-Wilk test can be seen in Table 1. There are many outliers in the feature variables, outliers can be seen in Figure 2. The Shapiro-Wilk test has been used to check if the all the feature variables are normally distributed. The p-value for each feature value is zero or near zero, indicating non-normal distributions for every feature variable.

Shapiro-Wilk Test for Feature Variables

Below are the feature variables and their associated p-values generated from the Shapiro-Wilk test.

P-Values for Shapiro-Wilk Test

Table 1.

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	0	0	0	0	0	0	0	0	0

V10	V11	V12	V13	V14	V15	V16	V17	V18	V19
0	0	0	0	0	0	0	0	0	0

V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
0	0	0	0	0	0	0	0	0	0

Data Samples

Below are two transactions in the dataset. One transaction is a genuine charge and the other is a fraudulent charge. Only ten feature variables are shown for the two samples.

Sample 1.

Class	'Time'	'V1'	'V2'	'V3'	'V4'	'V5'	'V6'	'V7'	'V8'	'V9'
Fraudulent	41991.0	-4.56	3.35	-4.57	3.61	-2.49	-1.09	-5.55	0.44	-2.42

Sample 2.

Class	'Time'	'V1'	'V2'	'V3'	'V4'	'V5'	'V6'	'V7'	'V8'	'V9'
Genuine	28418.00	1.56	-0.94	0.24	-1.44	-1.30	-0.72	-0.92	-0.21	-1.80

Average 'Amount' for Fraudulent and Genuine Charges

Table 2.

Class	Average 'Amount'	Max 'Amount'
Fraudulent	122.21	2125.87
Genuine	88.29	25,691.16

Exploratory Visualization

The Figures 1. – 12., show key characteristics of the dataset. Histograms, QQ-plots, scatter plots, and distributions show the structure of the data and the differentiation of fraudulent and genuine charges.

Distribution of the 'V14' variable

The 'V14' variable is extremely left-skewed. Most of the values are clustered around the mean which is near zero. However, there exist values that are extremely distant from the mean value of 0. Figure 2 highlights some of these extreme outliers. More detailed visualizations such as Figure 3 will show whether these extreme values are indicative of fraudulent charges.

'V14' Statistics

Table 3.

Class	Average 'V14'	Variance 'V14'
Fraudulent	-6.97	18.27
Genuine	0.01	0.8

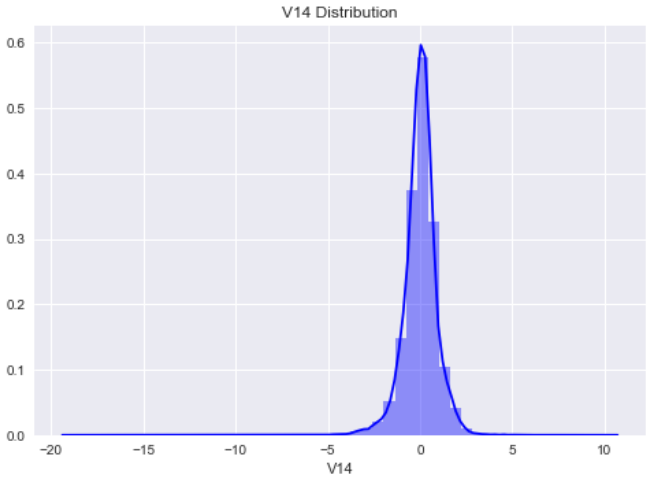


Figure 1.

Visual of extreme outliers in 'V14'

The vertical orange bar is placed at three standard deviations below the mean value of 'V14'. The vertical red bar is placed at the minimum value of 'V14', this value is roughly -19.2 which is considered an extreme outlier.

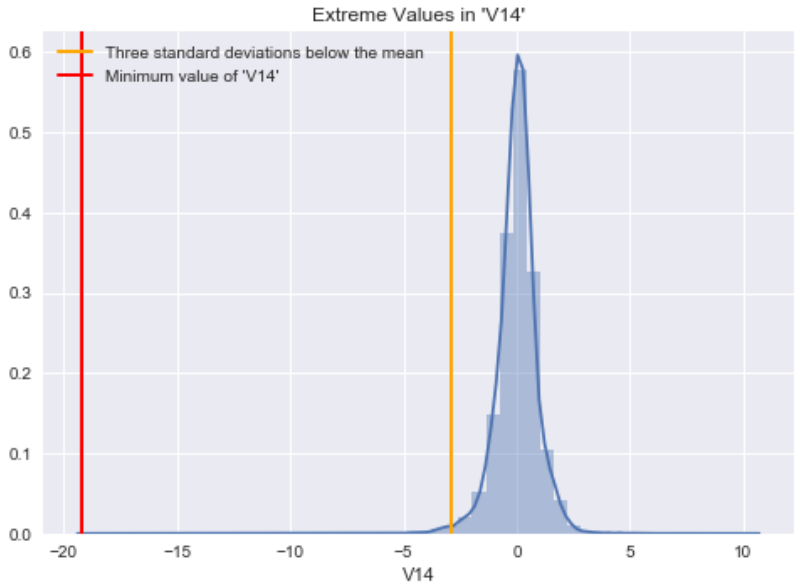


Figure 2.

There are many outliers in the feature variables. In this case, the outliers are helpful for classifying fraudulent and genuine charges. Figure 3 illustrates this point, the distribution for the variable 'V14' is plotted for Class 1 and Class 0. The extreme values that Class 1 take relative to Class 0 are beneficial for the task of classification. Gaussian mixture models for clustering operate on the principle that separate classes come from separate distributions. When clustering with a Gaussian Mixture model, each data point has a probability of belonging to one of k distributions, and it is assigned to one of the k distributions that it has the highest probability of belonging to. The distributions of the 'V14' variable for the separate classes are very different. The distribution of the 'V14' variable for fraudulent charges has a much higher variance than the distribution of 'V14' for genuine charges. Table 4. provides the mean and variance values for the 'V14' variable for fraudulent and genuine charges.

Table 4.

Class	Mean	Variance
Genuine	0.01	0.8
Fraudulent	-6.97	18.27

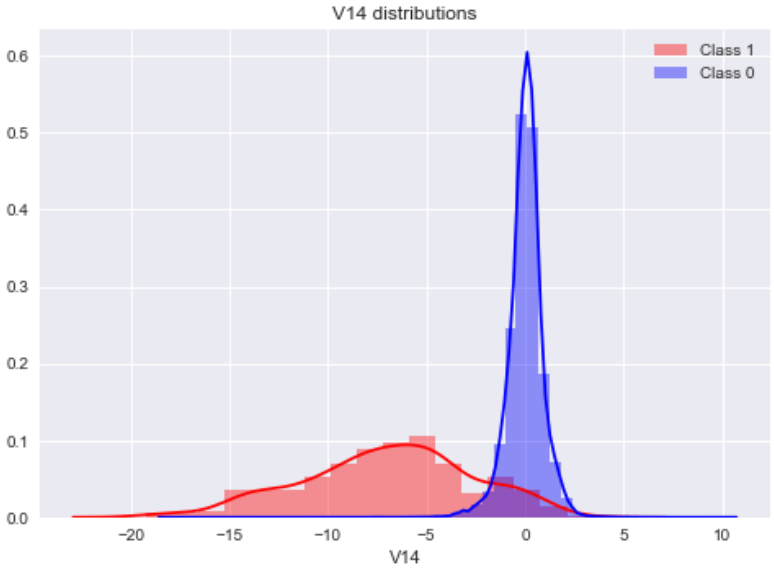


Figure 3.

Figure 4. Data that is normally distributed would closely track the red line in the QQ-plot. The values for 'Amount' are extremely far from this line.

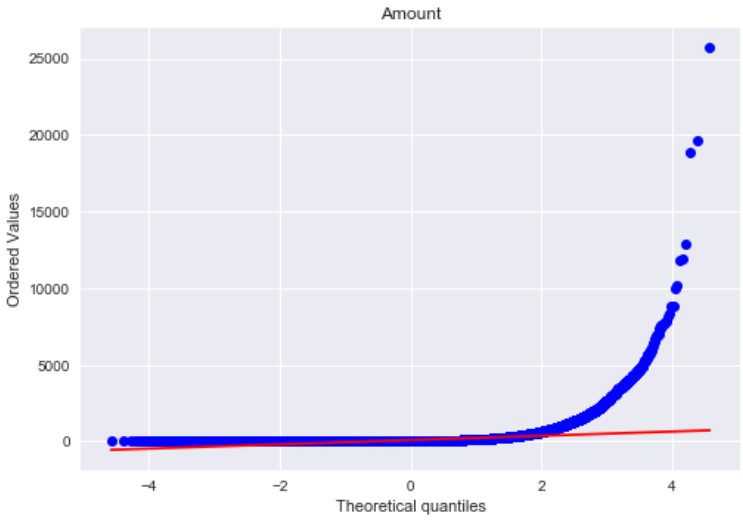
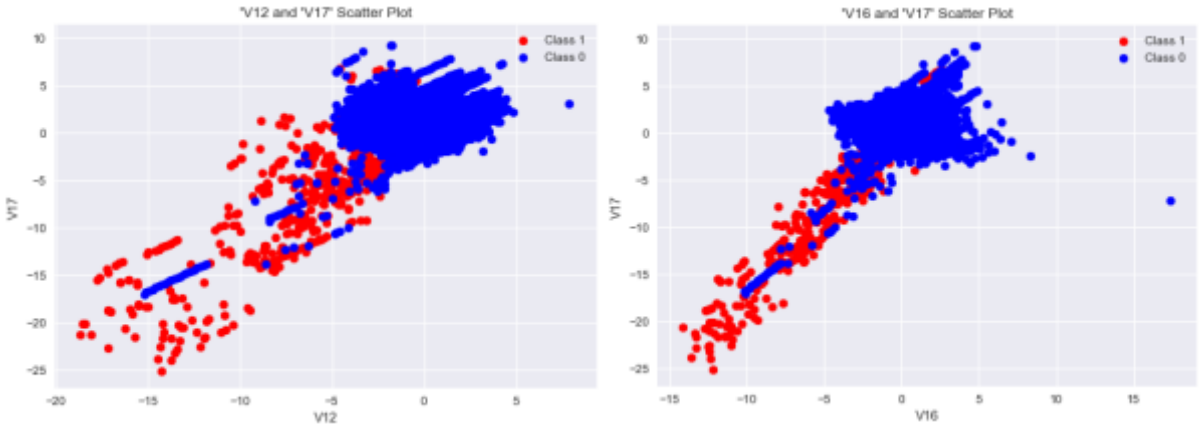


Figure 4.

Scatter Plots of Feature Pairs

Figure 5 shows a scatter plot that has a x-coordinate of 'V12' and a y-coordinate of 'V17'. Genuine charges are colored blue and fraudulent charges are colored red. Figure 6 is a scatter plot for the variables 'V16' and 'V17'. The clusters of class groups are highly visible in the scatter plots. This clustering provides intuition for how a model might classify fraudulent and genuine charges.



Figures 5. & 6.

Times of Purchases

Figures 7 - 9 show the frequency of charges through bar charts and a histogram. Sixty-seven percent of the charges occur in the second half of the day. The sixteenth hour has the highest frequency of charges. Below is a statistical summary of hours on a twenty-four-hour interval.

Table 5.

Mean	Standard Deviation	Min Value	25th Percentile	50th Percentile	75th Percentile	Max Value
13.86	5.743	0	10	14	18	23

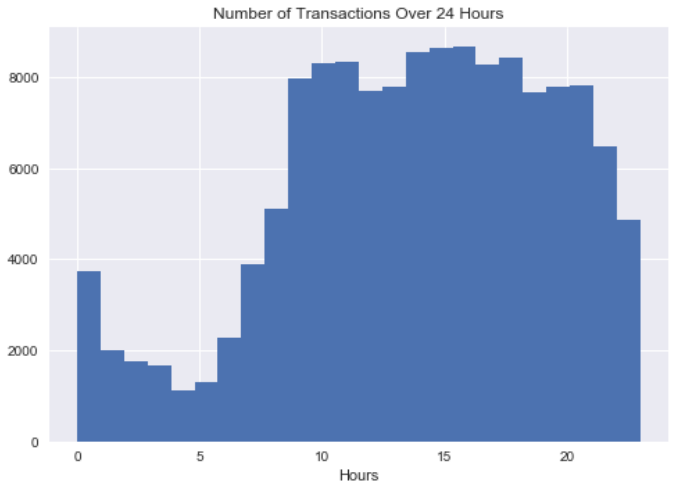


Figure 7.

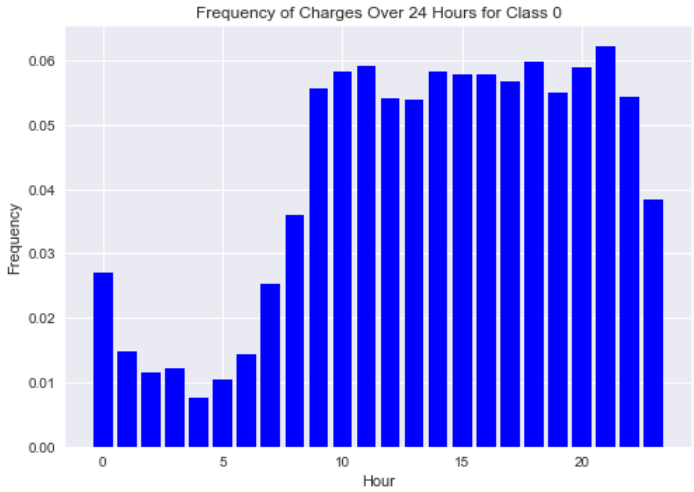


Figure 8.

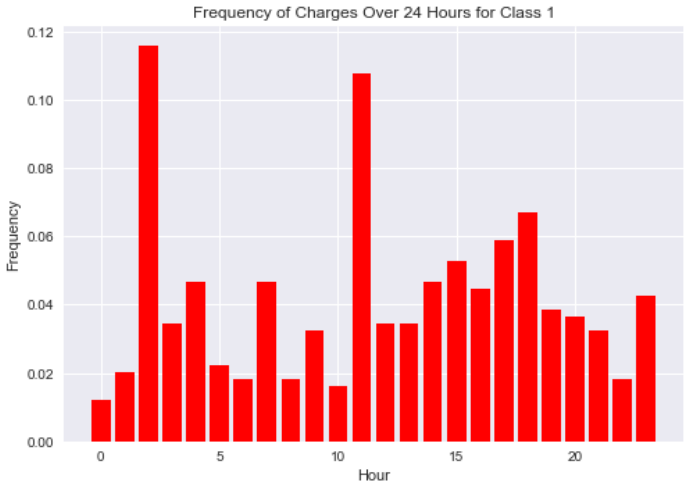


Figure 9.

Algorithms and Techniques

The decision tree classifier works by passing data through a series of nodes that ask questions about the data and depending on the answers at various nodes; the data will be placed in a leaf that represents a class label. For example, if the task is to classify animals, a tree could be constructed that has a node >= 15 feet tall, if the animal is >= 15 feet tall it will end up in a leaf of the tree that labels the animal 'Giraffe.' Many more nodes could be specified to split the animals into classes. The nodes are constructed to maximize information gain between the parent node and child nodes. Information gain measures the difference in impurity between the parent node and the child nodes. In the case of two classes, the impurity is highest when the class proportions of data points are both 0.50 at a given node. Impurity is lowest when a node only

contains data points that are from one class. For this problem specifically, the goal is to train a decision tree that can correctly classify charges in the test set. When initially training the model, the hyper-parameters for the Decision Tree classifier will not be modified from the default settings set by scikit-learn. Some of the hyper-parameters of a decision tree classifier include maximum depth and minimum samples per leaf. Higher levels of maximum depth can contribute to over fit or high variance models, lower values for minimum samples per leaf can also contribute to over fit models. The decision tree classifier is a non-parametric model and will be helpful in observing the difference between performance between parametric and non-parametric models.

The Support Vector Machine classifier separates data points in different classes with a linear separator, such that the margin between classes is maximized. Support vector machines can also separate classes that are not linearly separable in their current dimension. This is done by mapping the feature values to a higher dimension then separating the classes with a hyper-plane. Mapping feature variables to higher dimensions where they can be linearly separated is called the kernel trick. Using a SVM for this problem will be useful if the data is not linearly separable in its current dimension. Maximizing the margin of the hyperplane that separates data points is a key advantage that a SVM has over logistic regression. Maximizing the distance between the different classes can potentially make a model that generalizes better for new data. The initial SVM that is trained will have hyper-parameters that are at default settings. The kernel of the SVM is set as 'rbf,' this setting allows for non-linear separations through the kernel trick. The SVM may exceed the performance of logistic regression because of the position of the hyper-plane being optimized and the capability to separate data that is not linearly separable.

The objective of the logistic regression model is to linearly separate classes of data points. The parameters of the model are derived from the gradient descent algorithm. The gradient descent algorithm updates parameter weights to minimize the error function. The weights are iteratively updated by subtracting the gradient of the error function with respect to the weights from the current weight values, additionally, a learning rate alpha multiplies the gradient to control the size of the steps when reducing the error function. The relative simplicity of logistic regression makes it a good benchmark model for a more sophisticated model such as a SVM. If the SVM exceeds the performance of the logistic regression model, this could indicate that there are non-linear boundaries between classes of data points. The logistic regression model will have default hyper-parameter values set by scikit-learn. The parameter 'C' controls the penalty for the magnitude of weights, higher values of C penalize the magnitude of parameters less than low values of C. The value of C is tuned to control the complexity of the logistic regression model.

Principal Component Analysis is employed to reduce the dimensionality of data while losing as little information about the data as possible. The process of deriving principal components is carried out in the following way. A covariance matrix is calculated from the feature variable matrix, eigenvalues and eigenvectors are then derived from the covariance matrix. Each eigenvector has a corresponding eigenvalue and eigenvectors are ranked according to the magnitude of their eigenvalues. The rank of eigenvectors is based on the size of their eigenvalues, higher eigenvalues correspond to how much variability in the data each eigenvector explains. The eigenvectors form an orthogonal set of vectors which represent the principal components. To find out where a data point lies on a principal component the following is

carried out. A data point such as [x, y, z] is mapped to a position on a principle component by the average value of each variable being subtracted from the values in the vector e.g. [x - mean(X), y - mean(Y), z – mean(Z)] then the dot product of this vector and a principal component calculated. This results in a scalar value on a principal component axis. Principal component analysis will be used in this to reduce the dimension of the feature variables to allow for better visualization of the structure in the data. Here is a link that has a nice visual presentation of principal component analysis. http://setosa.io/ev/principal-component-analysis/

Benchmark

The benchmarks consist of a naive prediction accuracy score and the accuracy, precision, recall and F1 score of a logistic regression model. The accuracy attained by the naive prediction of assuming all charges are genuine is (284807 - 492) / 284803, this value is roughly 0.99827. Since the logistic regression model is a relatively simple classification model, the metrics that the logistic regression model generates on the test set will be appropriate benchmarks for a more sophisticated model such as a SVM to be measured against.

Data Preprocessing

To extract the features that are most predictive for classifying a charge as fraudulent or not, the feature_importances method of a random forest classifier will be employed. This will reduce the number of features in model training and will reduce training time. Feature importance corresponds to how much splitting on a feature in a forest of trees decreases impurity. The feature that corresponds to the greatest decrease in impurity on average is the most important feature.

After training a Random Forest Classifier with 100 trees, the most predictive features have been returned with the feature_importances method. The six most predictive feature variables consist of 'V10', 'V11', 'V12', 'V14', 'V16', 'V17'. These six features will be used to train the classification models. Figure 10. displays a bar chart that shows feature importance. As seen in Figure 10., ‘V12’ is the most important feature as it has the highest value.

The feature variables 'V10', 'V11', 'V12', 'V14', 'V16', 'V17' are already standardized, so it is unnecessary to manipulate these feature variables. The variables 'V10', 'V11', 'V12', 'V14', 'V16', 'V17' all have a mean value of zero or near zero which makes standardizing unnecessary. To illustrate standardization more concretely below is an example of pseudocode that standardizes a hypothetical variable X.

Standardization Pseudocode

```
Let X[i] be the ith variable that is a member of X, let 'sigma' be the standard deviation of X and let 'm' be
the mean of X

for i in range(len(X)):

        X[i] = (X[i] - m) / sigma
```

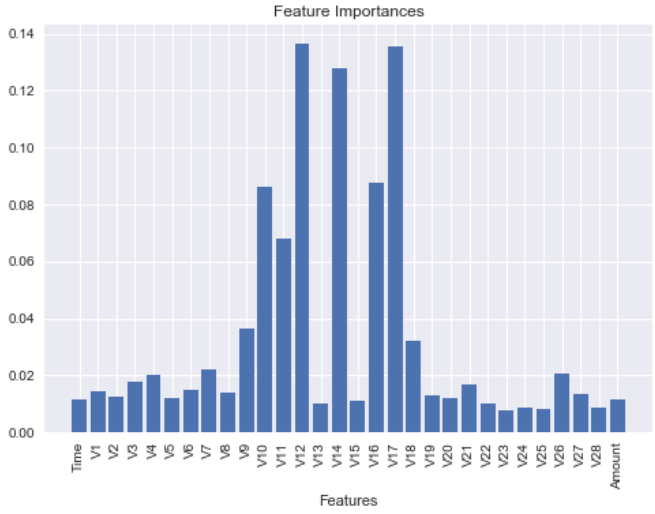


Figure 10.

Implementation

The feature and target variables have been split into training and testing sets with the "train_test_split" function. The testing set is 20 % of the data; the test set is not used for model training. A SVM classifier, logistic regression, and decision tree classifier were instantiated and fit to the training set. The parameters of the SVM and logistic regression models are learned through the gradient descent algorithm during training. The gradient descent algorithm adjusts the parameters to minimize the error function of these parameterized models. The decision tree classifier turns into a structure of nodes and leaves, what leaf a data point falls into depends on the values of the feature variables. Maximizing information gain is what determines splits in the tree. After model training, the models made predictions on the test set. The metrics that the models produced are shown in Table 6.

Model Performance

Table 6.

Model	Accuracy	Precision	Recall	F1
Logistic Regression	0.9991	0.8787	0.5742	0.6946
Support Vector Machine	0.9995	0.9425	0.8118	0.8723
Decision Tree Classifier	0.9992	0.7809	0.8118	0.7961

As seen in Table 6., the Logistic Regression, Support Vector Machine, and Decision Tree classifier achieved accuracy scores on the test set that exceed the naive accuracy benchmark. The

SVM and logistic regression models have strong precision scores; this indicates that these models are not making excessive false positive predictions. Observing Table 6., it is probable that identifying all the fraudulent charges in the test set is difficult; this is indicated by relatively weak recall scores when compared to precision scores. Overall the support vector machine had the strongest performance generating a F1 score of 0.8723; this is below the target of >= .9. With hyper-parameter tuning the SVM may achieve the mark of F1 >= 0.90. The Support Vector Machine will be further optimized to increase its F1 score.

Refinement

Given that the Support Vector Machine recorded the strongest performance, the SVM's hyper-parameters will be tuned using k - fold cross-validation and grid search. Selecting optimal hyper-parameters using k-fold cross-validation and grid search is done in the following way. The training set is partitioned into k-folds, k - 1 folds are used for training and 1 - fold is used validation. All combinations of the hyper-parameters are used when training on the k - 1 folds, combinations of hyper-parameters are generated by taking the Cartesian product of the hyper-parameter grid. A model that corresponds to each hyper-parameter combination is then tested on the validation set. The average score on the validation set is recorded for each model with different hyper-parameters, the hyper-parameter combination is selected based on the highest average score on the validation set.

Table 7. displays the hyper-parameter grid that will be used during k – fold cross-validation. A total of nine hyper-parameter combinations will be tested during the cross-validation process. The value of k will be set to five, and the scoring metric on the validation set is set to F1. The default hyper-parameters are in the grid to ensure that any selected hyper-parameters match or exceed the performance of the original SVM.

Parameter Grid

Table 7.

‘Kernel’	**‘gamma’**	**‘C’**
[‘rbf’]	[.01, 1/6, 0.5]	[1, 5, 10]

After running the process of k – fold cross-validation and grid search, the optimal hyper parameters are ‘kernel’ = ‘rbf’, ‘C’ = 1, ‘gamma’ = 1/6. The set of optimal hyper parameters are the same as the default hyper parameters. The fact that the hyper parameters did not change also means that model performance did not change. Tables 8. and 9. show the performance metrics of the optimized and un-optimized SVM.

Un-Optimized SVM Performance

Table 8.

Model	Accuracy	Precision	Recall	F1
Un-Optimized Support Vector Machine	0.9995	0.9425	0.8118	0.8723

Optimized SVM Performance

Table 9.

Model	Accuracy	Precision	Recall	F1
Optimized Support Vector Machine	0.9995	0.9425	0.8118	0.8723

Final Hyper Parameter Values

Table 10.

'Kernel'	'gamma'	'C'
'rbf'	1/6	1

The objective of a >=0.90 F1 score was not achieved. To stress test the optimized model, predictions were made exclusively on the first day of charges. The first day's charges were split into training and testing sets using the train_test_split function. The optimal SVM's test set performance is recorded in Table 11. This exercise is to test how well the model performs on a different subset of data. The performance of the model is stronger when tested on the first day's transactions, achieving a F1 score of 0.9166.

Optimal SVM Performance for First Day Charges

Table 11.

Model	Accuracy	Precision	Recall	F1
Optimized Support Vector Machine	0.9996	0.9821	0.8593	0.9166

Model Evaluation and Validation

After the process of training three models and optimizing the Support Vector Machine, the Support Vector Machine hyper-parameters C = 1 and gamma = 1/6 delivered the strongest results on the test set. The objective of F1 >= 0.90 was difficult to achieve since the recall score

of 0.8118 pulled the F1 score down. The precision score of 0.9425, indicates that when the SVM makes a fraudulent classification, there is a solid chance it is fraudulent.

The parameters of C = 1, and gamma = 1/6 were the chosen hyper-parameters after running the grid search and k – fold cross-validation algorithm. The C value controls the penalty for misclassified data points, higher values of C correspond to more severe punishment for misclassification. The gamma parameter controls the range of influence that a data point has regarding separation boundaries, higher values of gamma increase model variance. The hyper-parameters of the optimized SVM are appropriate given they were selected with the combination of grid search and cross-validation. The values of gamma and C are also relatively low indicating that the model is not over fit.

The final metrics that the optimized model generated exceeded the naïve accuracy score and logistic regression metrics. Model robustness was tested by making predictions on exclusively the first day's charges, the results are in Table 11. When predicting exclusively the first day's charges, the F1 score came in at 0.9166 on the test set, which is a stronger score than on the entire testing set. This is evidence that the SVM model is consistent when making predictions on different subsets of data. The results of the model can be trusted due to several reasons. The first reason is that predictive features were chosen to train the SVM. The features and targets were split into training, and testing sets and three models were trained with this data. Three models made predictions on the test set, and the strongest performing model (SVM) was chosen to be optimized. The SVM was optimized by finding optimal hyper-parameters with grid search and cross-validation. The optimized model was then trained with the entire training set and then made predictions on the test set. The scores were strong on the test set, exceeding all benchmarks and returning a F1 score of 0.8723

Justification

The benchmark accuracy of ~ 0.99827 was surpassed by the optimized SVM model, with the optimized SVM model achieving an accuracy score of 0.9995 on the test set. The accuracy, recall, precision, and F1 score of the logistic regression model was surpassed by the optimized SVM model. This is seen comparing Table 12. and Table 13. The final model exceeded all benchmarks except but did not reach the goal of a F1 score >= 0.90 on the entire test set. The optimized SVM achieved a F1 score of 0.9166 on the test set that only includes charges on the first day. With the relative strength of the optimized SVM, the initial problem is mostly solved.

Logistic Regression Performance

Table 12.

Model	Accuracy	Precision	Recall	F1
Logistic Regression	0.9991	0.8787	0.5574	0.6946

Optimized SVM Performance

Table 13.

Model	Accuracy	Precision	Recall	F1
Optimized Support Vector Machine	0.9995	0.9425	0.8118	0.8723

Free-Form Visualization

To create Figure 11, the features were reduced from six dimensions to three with principal component analysis. Fraudulent charges are colored red and genuine charges are colored blue. The placement of the different classes in three-dimensional space allows for further visualization of the difference between genuine and fraudulent charges. It is also seen that there is not a clean break from fraudulent charges and legitimate charges in the three dimensions. This is likely a reason why it was difficult to attain a high recall score while simultaneously maintaining a high precision score.

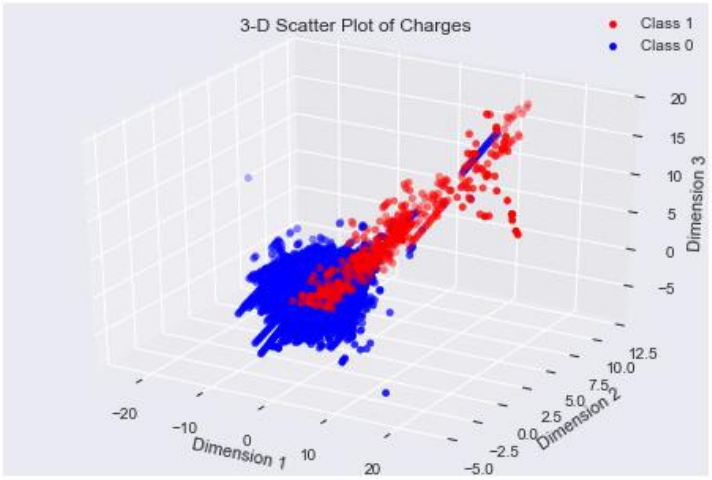


Figure 11.

Benford's Law

The frequency of leading digits for many real-world data sets closely follow the distribution predicted by Benford's law. Benford's law predicts that the leading digit should have the probability log10(1 + 1/d) of occurring in the data set, where d is the leading digit. The frequency for each leading digit 1 - 9 for both fraudulent and non-fraudulent charges has been calculated and compared to the Benford's law distribution. (Wolfram MathWorld, n.d.)

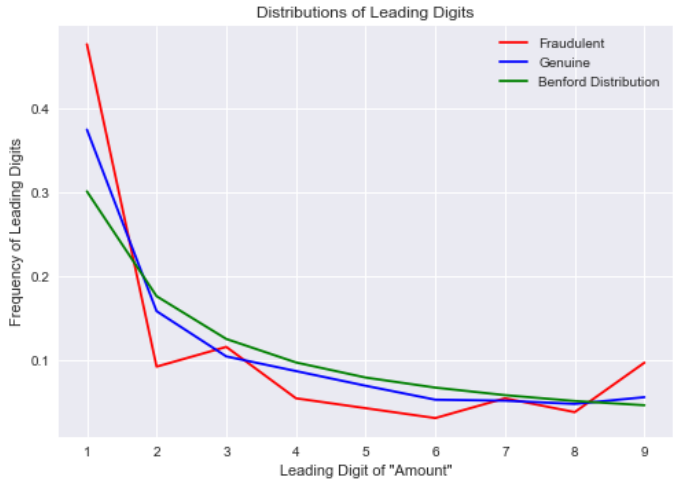


Figure 12.

Both fraudulent and genuine leading digits closely track the predictions made by Benford's law. Leading digits of genuine charges are closer to the predictions made by Benford's law than the leading digits of fraudulent charges. For fraudulent charges, the probability of a leading digit being 9 or 1 is noticeably different than the probability predicted by Benford's law. The probability of the first digit being a 9 for the fraudulent charges is ~ 9.6 percent versus ~5.5 percent for a non-fraudulent charge.

Reflection

Many techniques were employed to find a signal in the initially large number of feature variables. To find patterns in the dataset the most predictive features were selected for model development. After training a few algorithms, the SVM algorithm generated the strongest results in the initial phase of model development and was selected for further refinement. The process of refinement involved finding optimal hyper parameters with grid search and k-fold cross-validation. The parameters associated with the highest performing F1 score on the validation set were then chosen for the Support Vector Machine. Hyper-parameter tuning did not improve the initial SVM model's F1 score of 0.8723, although the SVM did exceed all benchmarks.
There were several difficult aspects of this project, some of the difficulties include the imbalance in the data set and the lack of a clean delineation between fraudulent and genuine charges. The imbalance in the dataset contributed to the fact that there was much more information about the structure of genuine charges than fraudulent charges. Had there been more fraudulent charges, a

better-defined pattern could have emerged and contributed to better model performance. Attaining a F1 score of >= 0.90 on the test set was difficult due to the recall score dragging on performance. The 0.8723 F1 score of the SVM is a respectable score for this problem and provides evidence that this type of problem can be solved with Support Vector Machine models.

Improvement

There are several techniques that can possibly improve on the results attained by the SVM model. Having more values for C and gamma in the parameter grid could have possibly returned better C and gamma values. This can be implemented if there is significant time to allow for training, or there is very powerful computational power available for model training. Another model could have been optimized such as the decision tree classifier. It is possible the optimized decision tree classifier could have achieved better results than the optimized SVM. Additional models that could have been tested include random forests and deep neural networks. Several techniques exist that can alleviate the problems associated with unbalanced datasets. These techniques involve under sampling the most frequent class or oversampling the less frequent class which would lead to a more balanced dataset. There is also a technique called SMOTE that generates synthetic data points of the minority class to alleviate class imbalance (Microsoft, n.d.). If the final solution F1 = 0.8723 is used as a benchmark for future research, there is a very strong likelihood that it could be surpassed. The techniques I have mentioned such as more extensive model tuning and special sampling techniques for imbalanced datasets are areas to explore that could yield results that surpass the performance of the Support Vector Machine.

References

CBINSIGHTS, Startups Using Machine Learning And Behavioral Biometrics To Fight Fraud. (2017, September 14). Retrieved October 25, 2017, from https://www.cbinsights.com/research/fraud-prevention-machine-learning-biometric-startups/

Dal Pozzolo Andrea, Caelen Olivier, Johnson R and Bontempi Gianluca . Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

Economist, (2014) Skimming off the top, Retrieved October 24, 2017, from https://www.economist.com/news/finance-and-economics/21596547-why-america-has-such-high-rate-payment-card-fraud-skimming-top

Microsoft (n.d.). Retrieved October 25, 2017, from https://msdn.microsoft.com/en-us/library/azure/dn913076.aspx

Wolfram MathWorld, (n.d), Benford's Law. Retrieved October 25, 2017, from http://mathworld.wolfram.com/BenfordsLaw.html