Project Report - Telco (Project 1)

Cherie Anderson, Madeline Frasca, Song Gao, Matthew Hodde, Katie Maurer

IDIS 450

Group 20

Executive Summary

The overall purpose of this data analytics project is to improve the churn rate at Telco, a company offering a variety of services and packages. These various services and packages include phone capabilities, streaming, internet, security, backup, and technical support. Telco was trying to determine the current reasons why their customers churn and how to avoid these reasons in the future. Customer churn, which is the focus on the project, is best defined as how often customers cancel their services at a given company. The team was provided with a dataset including a multitude of customer information, which services they chose, and whether or not they churned.

For the data to be interpreted by a computer, there were many steps needing to be taken to transfer the raw data into a more consumable format. This began with cleaning the data to identify/fix any missing values, eliminating non-significant data columns, and detecting outliers. In addition, the team utilized dummy variables to convert categorical variables into numerical values that could be later used in the necessary calculations. After this task was completed, Power BI was used to create exploratory data analysis (EDA) graphics to better visualize and interpret the data. Lastly, the group determined which classification models should be built to evaluate the data.

The team chose to build the following models: Linear Regression, Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), and K-Nearest Neighbor (KNN). Each of these models produced specific performance metrics such as an accuracy score, confusion matrix, or classification report. Based on these measurements, the team determined that SVM was the best model to depict the data because it had the best accuracy score. A detailed description of each of these models can be found in the body of this report.

The findings of this project can be attributed to the diligent work of each team member. The team split the assignment's tasks into individual responsibilities. Cherie was in charge of creating the EDA graphics. Katie took the lead role in cleaning the data. Matthew and Madeline split responsibility for creating the classification models based on the cleaned data. Lastly, Song analyzed the overall findings to determine the optimal model and make significant conclusions. Overall, each member worked together to create the final deliverables such as the PowerPoint and final report.

Exploratory Data Analysis (EDA)

EDA was a critical task the team performed in Power BI for the purpose of data visualization. The goal was to explore any initial relevant insights about the data set regarding significant variables impacting Telco's churn rate. The four variables compared to customer churn were customer tenure, senior citizen status, gender, and average monthly charges. All of these graphics were created and attached below including descriptions of what insights were identified within each.

Within the EDA comparing tenure to churn, it was found that those with the lowest and highest tenures are those with the greatest churn rates. Those with the lowest tenures are identified as the most recent customers, which can be concluded logically as well as through data analytics. Those with the longest tenures seem to be the most loyal customers. However, there is a drastic difference between the lowest and highest tenures compared to the rest of the tenure rates. Therefore, this should be investigated further through the team's code and Telco's future actions.

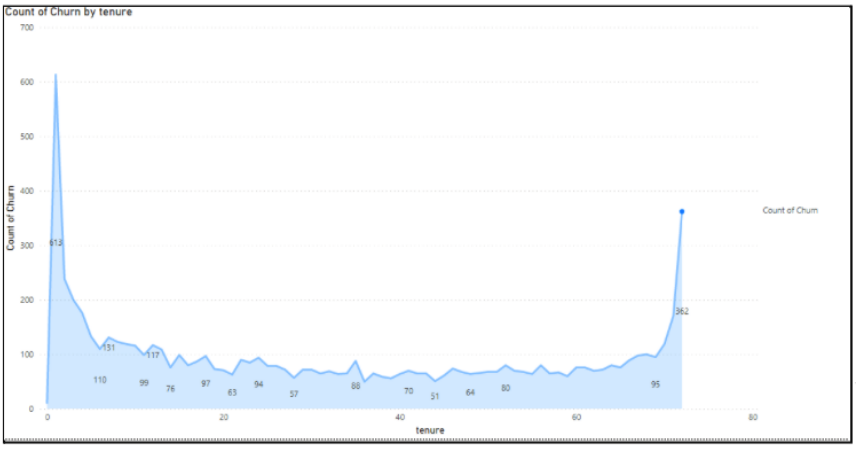


Figure 1: Tenure Analysis

Next, *Figure 2* depicts the count of senior citizens out of the total number of customers who churn. It can be seen that the majority of customers who churn are not senior citizens. This means 83.79% of Telco's churning customers are below the age of 65 years old. Based on this finding, Telco can use best practices for targeting and retaining the younger generation of customers since it appears that senior citizens are less likely to leave Telco.

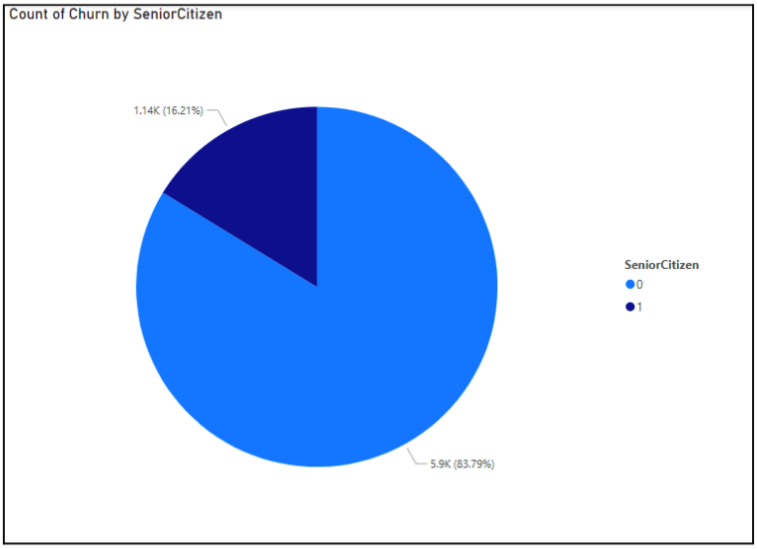


Figure 2: Senior Citizen Analysis

Through the analysis of *Figure 3*, since the split between males and females is extremely close to one another in regards to the churn rates, it can be determined that gender does not affect churn rate. This information is valuable because Telco now recognizes gender specific customer ads or efforts will not be beneficial in comparison to potential other segmented efforts.

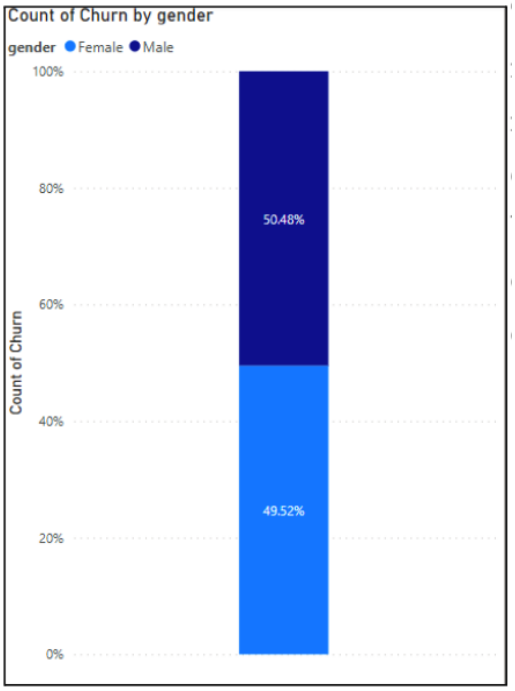


Figure 3: Gender Analysis

The last analysis the team conducted was on the average monthly charge amount (in dollars) vs the number of customers who churned, which can be seen above in *Figure 4*. It can be concluded that the highest number of customers churn at the monthly average amount of $20, with the second highest being at $25. In addition, there is likely a trend that the more dollar amount a customer spends per month, the less likely they are to churn. However, there is a slight increase in churn rate around the $75-85 range due to many potential reasons. Overall, each of these visuals helped the group better understand the type of data they were working with and any potential trends that exist.

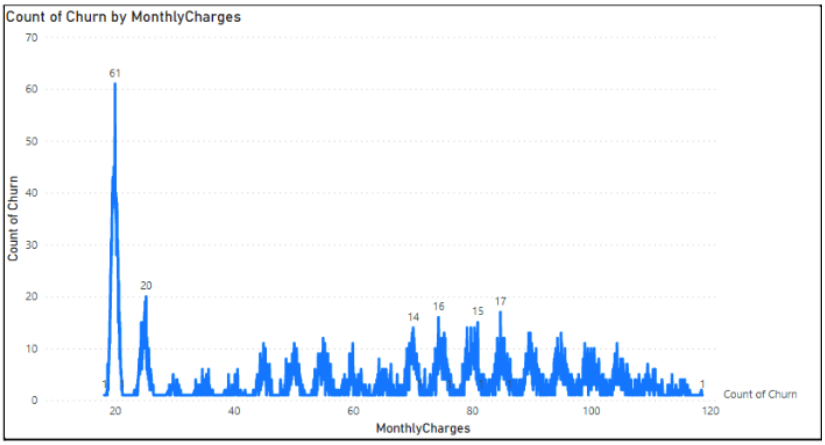


Figure 4: Monthly Charge Analysis

Technical Report of Support Vector Machine (SVM) Model

First, the team created a Support Vector Machine (SVM) Model to evaluate the data. To set up the SVM model a test size and random state are selected and test model parameters are entered. This provides a baseline for how the model is expected to behave. After a long run session, the accuracy score for SVM ended up being 0.8041 after an initial prediction of 0.8019 (which is shown in *Figure 4* below.)

```
[75] # SVM
     xTrain, xValid, yTrain, yValid = train_test_split(X, Y, test_size = 0.2, random_state = 1)

[76] SVModel = SVC(kernel = 'linear', C=10, gamma = 'auto')
     SVModel.fit(xTrain, yTrain)

     SVC(C=10, gamma='auto', kernel='linear')

[77] confusion_matrix(yTrain, SVModel.predict(xTrain))

     array([[3689,  424],
            [ 692,  829]])

[78] #accuracy score SVM for Y train
     accuracy_score(yTrain, SVModel.predict(xTrain))

     0.8019169329073482

[79] confusion_matrix(yValid, SVModel.predict(xValid))

     array([[936, 125],
            [151, 197]])

[80] #accuracy score SVM for y test
     accuracy_score(yValid, SVModel.predict(xValid))

     0.8041163946061036

[81] krn = ['linear', 'poly','rbf','sigmoid']
     rng_C = np.arange(1, 15, 5)
     rng_deg = np.arange(2, 5)

[82] param = {'kernel' :krn,
              'C' :rng_C,
             'degree' :rng_deg}
```

Figure 4: Code for SVM Model

The increase of accuracy score was due to having an optimized model from a param search. However, a larger C-value would have undoubtedly provided a more accurate score, but the time to completion would have been unreasonable. Next, the confusion matrix provided insight to the Type I and Type II errors in the model (as shown in *Figure 5*.)

```
[83] SVModel = SVC()
     GridS = GridSearchCV(SVModel,param, cv=5)
     GridS.fit(xTrain, yTrain)

     GridSearchCV(cv=5, estimator=SVC(),
                  param_grid={'C': array([ 1,  6, 11]), 'degree': array([2, 3, 4]),
                              'kernel': ['linear', 'poly', 'rbf', 'sigmoid']})

[84] GridS.best_params_

     {'C': 1, 'degree': 2, 'kernel': 'linear'}

[85] SVModel = SVC(kernel='linear', C=1, degree=2)
     SVModel.fit(xTrain,yTrain)
     accuracy_score(yValid, SVModel.predict(xValid))

     0.8048261178140526

[86] y_pred_test = SVModel.predict(xValid)
     matrix = confusion_matrix(yTrain, SVModel.predict(xTrain))
     sns.heatmap(matrix, annot = True, fmt='d')
     plt.title('Confusion Matrix')
     plt.xlabel('Predicted')
     plt.ylabel('True')
     print(classification_report(yValid, y_pred_test))

                  precision    recall  f1-score   support

               0       0.86      0.88      0.87      1061
               1       0.61      0.57      0.59       348

        accuracy                           0.80      1409
       macro avg       0.74      0.73      0.73      1409
    weighted avg       0.80      0.80      0.80      1409
```

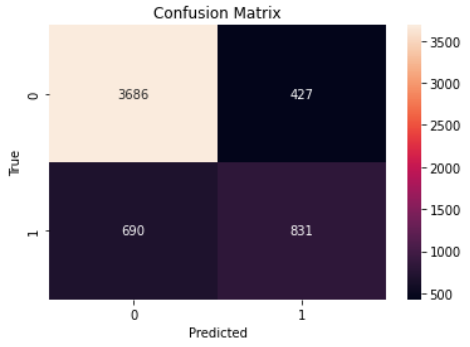


Figure 5: Code for SVM Model Continued

The Support Vector Machine model predicted 3686 no churn accounts, and 831 churn accounts. However, there were 690 Type I errors, and 427 Type II errors. The Type I error percentage is 690/5634 = 12.2% and a Type II error percentage of 422/5634 = 7.6%. Large data variables make the hyperplane more complex and each set results in recalculation of the data and more time used to get a final answer. The length of time required for SVM to run and vagueness of how the hyperplane is impacted makes this an inefficient model when compared to other methods that can be used.

Technical Report of Logistic Regression Model

Besides the Support Vector Machine models, the team members created the second model, being the Logistic Regression Model, in order to further evaluate the data. Ultimately, the goal of this task was to utilize the Logit function and find variables that could be used to predict and relate to a specific outcome (through binary classification.) To complete this task and conduct analysis, the team utilized the split training and validation data that was coded earlier in the project and fit the Logit model to the data. The OLS regression results from this model are seen below in *Figure 6*. Through these results, the team members were able to find the standard error, P values, and coefficients of each of the different columns and their variables. In addition, the r-squared calculation output was shown as being 0.286.

OLS Regression Results

Dep. Variable:	Churn	R-squared:	0.286
Model:	OLS	Adj. R-squared:	0.283
Method:	Least Squares	F-statistic:	101.9
Date:	Mon, 02 May 2022	Prob (F-statistic):	0.00
Time:	00:48:41	Log-Likelihood:	-2477.9
No. Observations:	5634	AIC:	5002.
Df Residuals:	5611	BIC:	5155.
Df Model:	22		
Covariance Type:	nonrobust		

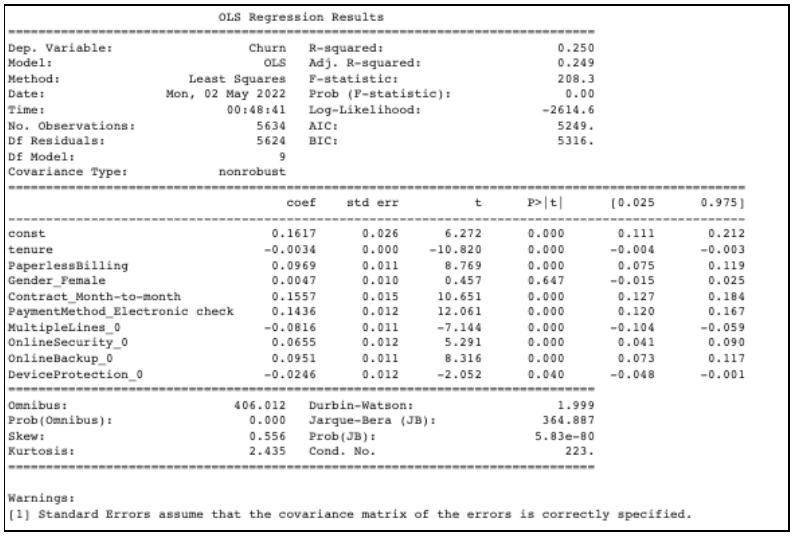
	coef	std err	t	P>|t|	[0.025	0.975]
SeniorCitizen	0.0360	0.015	2.465	0.014	0.007	0.065
Partner	-0.0115	0.012	-0.950	0.342	-0.035	0.012
Dependents	-0.0105	0.013	-0.813	0.416	-0.036	0.015
tenure	-0.0044	0.000	-12.870	0.000	-0.005	-0.004
PhoneService	-0.0136	0.096	-0.141	0.888	-0.202	0.175
TechSupport	-0.0609	0.027	-2.231	0.026	-0.114	-0.007
StreamingTV	0.0308	0.049	0.634	0.526	-0.064	0.126
StreamingMovies	0.0503	0.049	1.032	0.302	-0.045	0.146
PaperlessBilling	0.0520	0.011	4.623	0.000	0.030	0.074
MonthlyCharges	-0.0008	0.002	-0.354	0.723	-0.005	0.004
TotalCharges	-0.0008	0.002	-0.354	0.723	-0.005	0.004
Gender_Female	0.0645	0.025	2.566	0.010	0.015	0.114
Gender_Male	0.0616	0.025	2.446	0.014	0.012	0.111
InternetService_DSL	-0.0051	0.013	-0.389	0.697	-0.031	0.021
InternetService_Fiber optic	0.1784	0.107	1.673	0.094	-0.031	0.387
InternetService_No	-0.0471	0.047	-1.002	0.316	-0.139	0.045
Contract_Month-to-month	0.0942	0.019	4.974	0.000	0.057	0.131
Contract_One year	-0.0073	0.019	-0.389	0.697	-0.044	0.030
Contract_Two year	0.0393	0.020	2.000	0.046	0.001	0.078
PaymentMethod_Bank transfer (automatic)	0.0197	0.016	1.266	0.205	-0.011	0.050
PaymentMethod_Credit card (automatic)	0.0099	0.016	0.632	0.527	-0.021	0.041
PaymentMethod_Electronic check	0.0991	0.015	6.491	0.000	0.069	0.129
PaymentMethod_Mailed check	-0.0026	0.016	-0.163	0.870	-0.034	0.028
MultipleLines_0	0.0419	0.014	3.025	0.003	0.015	0.069
MultipleLines_1	0.0842	0.037	2.269	0.023	0.011	0.157
OnlineSecurity_0	0.0952	0.014	6.586	0.000	0.067	0.124
OnlineSecurity_1	0.0309	0.037	0.836	0.403	-0.042	0.103
OnlineBackup_0	0.1035	0.036	2.841	0.005	0.032	0.175
OnlineBackup_1	0.0697	0.060	1.164	0.244	-0.048	0.187
OnlineBackup_No internet service	-0.0471	0.047	-1.002	0.316	-0.139	0.045
DeviceProtection_0	0.0702	0.014	4.894	0.000	0.042	0.098
DeviceProtection_1	0.0559	0.037	1.510	0.131	-0.017	0.128

Omnibus:	303.151	Durbin-Watson:	2.006
Prob(Omnibus):	0.000	Jarque-Bera (JB):	287.812
Skew:	0.501	Prob(JB):	3.18e-63
Kurtosis:	2.527	Cond. No.	1.12e+16

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 5.01e-25. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Figure 6: OLS Regression Results

Since only the P values below 0.05 are desired to be analyzed, a new OLS regression results table was created, as seen on the next page, to show only the columns the team wanted to focus on primarily.



OLS Regression Results

Dep. Variable:	Churn	R-squared:	0.250
Model:	OLS	Adj. R-squared:	0.249
Method:	Least Squares	F-statistic:	208.3
Date:	Mon, 02 May 2022	Prob (F-statistic):	0.00
Time:	00:48:41	Log-Likelihood:	-2614.6
No. Observations:	5634	AIC:	5249.
Df Residuals:	5624	BIC:	5316.
Df Model:	9		
Covariance Type:	nonrobust		

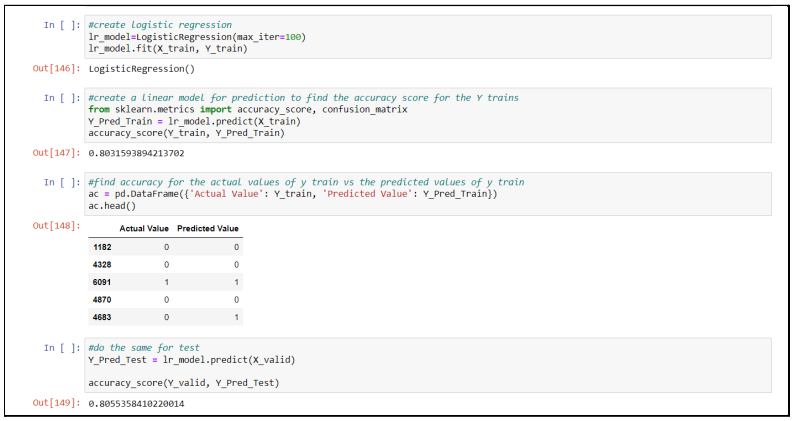
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1617	0.026	6.272	0.000	0.111	0.212
tenure	-0.0034	0.000	-10.820	0.000	-0.004	-0.003
PaperlessBilling	0.0969	0.011	8.769	0.000	0.075	0.119
Gender_Female	0.0047	0.010	0.457	0.647	-0.015	0.025
Contract_Month-to-month	0.1557	0.015	10.651	0.000	0.127	0.184
PaymentMethod_Electronic check	0.1436	0.012	12.061	0.000	0.120	0.167
MultipleLines_0	-0.0816	0.011	-7.144	0.000	-0.104	-0.059
OnlineSecurity_0	0.0655	0.012	5.291	0.000	0.041	0.090
OnlineBackup_0	0.0951	0.011	8.316	0.000	0.073	0.117
DeviceProtection_0	-0.0246	0.012	-2.052	0.040	-0.048	-0.001

Omnibus:	406.012	Durbin-Watson:	1.999
Prob(Omnibus):	0.000	Jarque-Bera (JB):	364.887
Skew:	0.556	Prob(JB):	5.83e-80
Kurtosis:	2.435	Cond. No.	223.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Figure 7: OLS Regression Results $P < 0.05$

After creating these results, the coefficient relationships can be identified between two different variables. For example, there is a negative coefficient between tenure and churn, which is identified at -0.0034. These trends were viewed and analyzed for all of the remaining significant columns. The data was then split into training and testing data, assigning 80% of the data as “train” and the random state at 10.



In [ ]:
```
#create logistic regression
lr_model=LogisticRegression(max_iter=100)
lr_model.fit(X_train, Y_train)
```

Out[146]: LogisticRegression()

In [ ]:
```
#create a linear model for prediction to find the accuracy score for the Y trains
from sklearn.metrics import accuracy_score, confusion_matrix
Y_Pred_Train = lr_model.predict(X_train)
accuracy_score(Y_train, Y_Pred_Train)
```

Out[147]: 0.8031593894213702

In [ ]:
```
#find accuracy for the actual values of y train vs the predicted values of y train
ac = pd.DataFrame({'Actual Value': Y_train, 'Predicted Value': Y_Pred_Train})
ac.head()
```

Out[148]:

	Actual Value	Predicted Value
1182	0	0
4328	0	0
6091	1	1
4870	0	0
4683	0	1

In [ ]:
```
#do the same for test
Y_Pred_Test = lr_model.predict(X_valid)

accuracy_score(Y_valid, Y_Pred_Test)
```

Out[149]: 0.8055358410220014

Figure 8: Logistic Regression Accuracy Score

The first observation to be made is that the logistic regression model has an accuracy score of 0.803 for the training data set, which means it is about 80% accurate. It is important to note that this is lower than the accuracy score for the support vector machine model (ran in the previous section.) However, it is still an overall good evaluation of the data since it has a higher score than other models that were created. Therefore, we decided to explain this model as one of the best three options.

Technical Report of K-Nearest Neighbor (KNN) Model

Lastly, the team created a K-Nearest Neighbor (KNN) Model to evaluate the data. The overall purpose of a KNN model is to create groups within the data determined by their proximity to each other. The idea behind this model is that variables in proximity to one another are similar. To complete the KNN task, the group first utilized the data preprocessing and splitting which was completed earlier in the lab. The next thing was to use the KNeighborsClassifier and metric functions to train the model and predict what the best k-score would be in order to have the most accurate data. Below in Figures 9 & 10 you can see the results of this as the error rate and accuracy score for k-scores between 1 - 40 were found and plotted. It was found that a k-score of 33 would be the best to use for the KNN model as the error rate at that point is 20.44% and the maximum accuracy score is 79.96%.

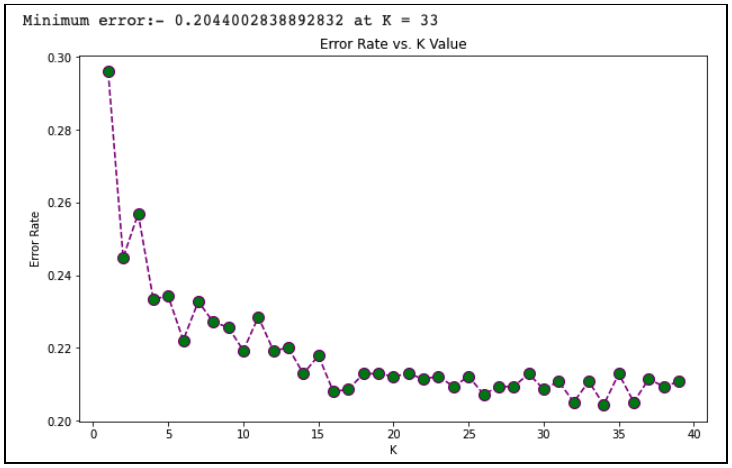


Figure 9: KNN Model of Error Rate vs K-Value

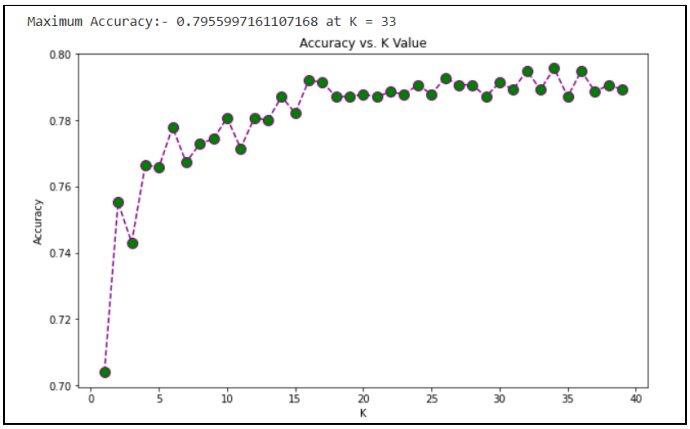


Figure 10: KNN Model of Accuracy Score vs K-Value

After these results were obtained, the next step was to create confusion matrices for the training and testing data. This is shown in Figures 11 & 12, these matrices were created by using the best parameters that were found with the GridSearch function. These results show that there were 3576 correctly predicted observations for the training data, and 900 for the testing data. However, there were 690 Type I and 427 Type II errors. The Type I error percentage for training is 626/3576 = 17.5% and a Type II error percentage of 537/3576 = 15%. The Type I error percentage for testing is 138/900 = 15.3% and a Type II error percentage of 161/900 = 17.9%. The reason for this is because when there are copious amounts of data, the error percentage will naturally increase. Overall, the total accuracy of these was predicted to be around 79% when using the classification report.

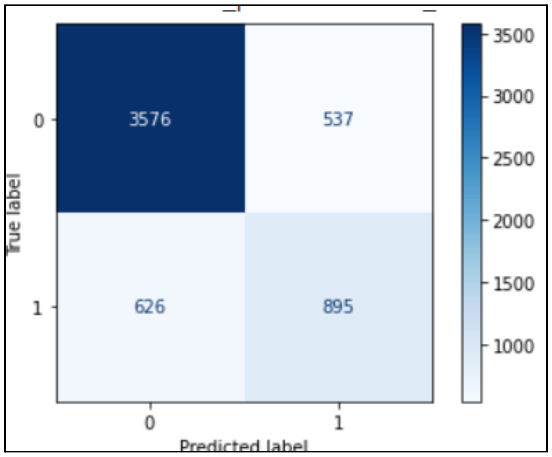


Figure 11: KNN Confusion Matrix for Training Data

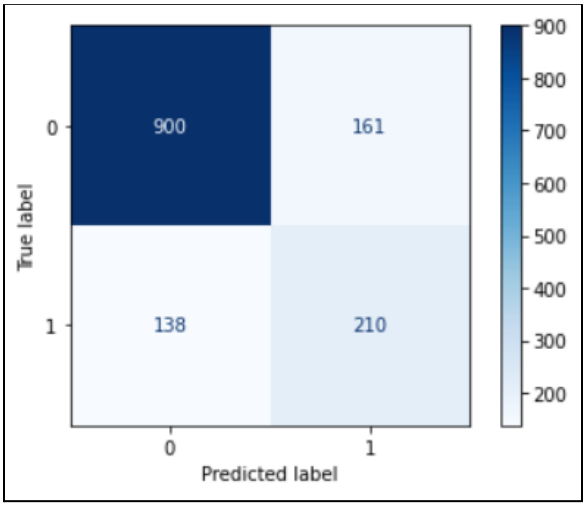


Figure 12: KNN Confusion Matrix for Testing Data

Comparison of Models and Best Model

Out of the three models that were evaluated, the one that was determined to be the best was the Support Vector Machine (SVM) model, we did this by comparing the accuracy scores of every model. The SVM model had an accuracy score of the predicted models being 80.41% accurate. The other two models had an accuracy score of 79.96 % for K-Nearest Neighbor (KNN) and an 80.3% accuracy for the Logistic Regression Model. Below in Figure 13 is a comparison of the accuracy of all three models. Overall, the reason why accuracy score was decided to be the determining factor for the "best" model was that it was the easiest way of comparison for all of the models. It would have been significantly harder to compare the confusion matrices of each model due to there being different type errors coming into play which would have added a different dimension to the comparison. Below in Figure 13 is a comparison of the accuracy of all three models.

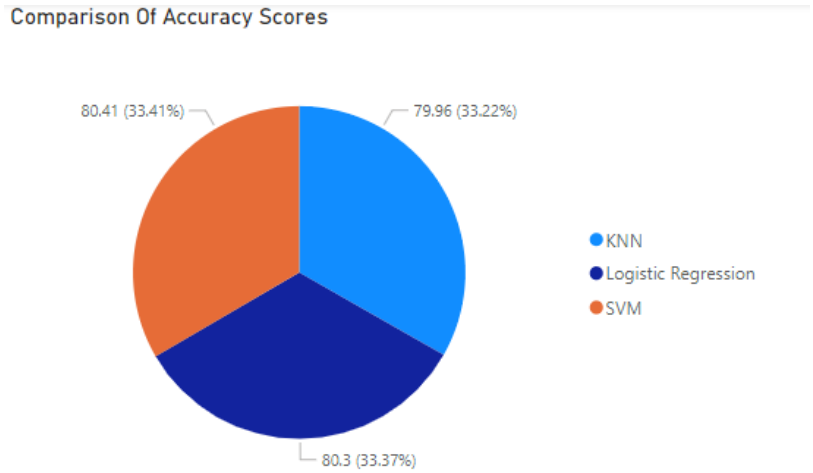


Figure 13: Comparison of Accuracy Scores

The reason why the SVM could have had the best accuracy score is due to the fact that the parameters were optimized for the accuracy score, unlike the other models which have a less complex way of calculating the scores with the parameters. Overall, the accuracy score of the SVM could have even been predicted to be higher if larger parameters were able to be used. The only reason why larger parameters were not used is due to the amount of time it takes to run SVM models. For just the parameters used, it took about 40 minutes to run the model, if the parameters were expanded it could have taken well over an hour. This also happens to be the downside of using the SVM methods because they take so long to run. The model that took the next longest time to run took about a third of the overall time.