Telco Data Analysis

Group 20
Project 1

Meet the Team

Matthew Hodde

Katie Maurer

Song Gao

Madeline Frasca

Cherie Anderson

Agenda

- Executive Summary
- Data Cleaning
- Exploratory Data Analysis
- Splitting & Standardization of the Data
- Technical Analysis of Data Classification Models
- Best Model & Conclusion

Executive Summary

- **Description:**

  As Telco is trying to keep their old customers while gaining new ones, they want to look at the churn rates (cancellation rates) and identify factors causing customers to leave to fix it.

- **Goals:**

  Ability to predict when a customer might leave to go the extra mile to ensure they change their minds

Data Cleaning Process

<u>Purpose:</u> Transform Data to remove errors, inconsistencies, and be usable

Data Cleaning Summarized

- Missing values
- Outlier detection & removal
- Categorical value to numerical
- Eliminated customer ID column (Irrelevant to outcome)
- Created dummy variables

Data Cleaning Code

```
#Ensure the object has the correct data
data.head(5)
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month

5 rows × 21 columns

Found:

- Shape
- Columns
- Dtypes
- Any Missing Values

```
#Determine the types of data
data.dtypes
```

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

Cont.

All "object" types were changed to 1s (Yes) and 0s (No)

```
#Replace every 'Yes' in data with a '1' and every 'No' with a '0'
data1.Partner.replace(('Yes', 'No'), (1,0), inplace=True)
data1.Dependents.replace(('Yes', 'No'), (1,0), inplace=True)
data1.PhoneService.replace(('Yes', 'No'), (1, 0), inplace=True)
data1.MultipleLines.replace(('Yes', 'No phone service', 'No'), (1,0,0), inplace=True)
data1.OnlineSecurity.replace(('Yes', 'No', 'No internet service'), (1, 0,0), inplace = True)
data1.OnlineBackup.replace(('Yes', 'No'), (1,0), inplace=True)
data1.DeviceProtection.replace(('Yes', 'No', 'No internet service'), (1,0,0), inplace=True)
data1.TechSupport.replace(('Yes', 'No', 'No internet service'), (1,0,0), inplace=True)
data1.StreamingTV.replace(('Yes', 'No', 'No internet service'), (1,0,0), inplace=True)
data1.StreamingMovies.replace(('Yes', 'No', 'No internet service'), (1,0,0), inplace = True)
data1.PaperlessBilling.replace(('Yes', 'No'), (1,0), inplace=True)
data1.Churn.replace(('Yes', 'No'), (1,0), inplace = True)
```

Dummies were created

```
# Split all Categorical Columns into Multiples
data2=data1.drop(['gender'],axis=1)
```

```
#Make a new column showing Female_Gender and Male_Gender
#Remove Original "Gender" column
dummy=pd.get_dummies(data1['gender'],prefix="Gender")
dummy
data2=pd.concat([data2,dummy], axis=1)
```

```
#Make a new column showing the different internet services as columns starting with "InternetService"
#Remove original column named "InternetService"
data3=data2.drop(['InternetService'],axis=1)
dummy=pd.get_dummies(data2["InternetService"],prefix="InternetService")
dummy
data3=pd.concat([data3,dummy],axis=1)
```

```
#Make a new column showing different contract types starting with "Contract"
#Remove original column named "Contract"
data4=data3.drop(['Contract'],axis=1)
dummy=pd.get_dummies(data3["Contract"],prefix="Contract")
dummy
data4=pd.concat([data4, dummy],axis=1)
```

```
#Do the same thing with "PaymentMethod" as we did for "Gender", "InternetServices", and "Contracts"
data5=data4.drop(['PaymentMethod'],axis=1)
dummy=pd.get_dummies(data4["PaymentMethod"],prefix="PaymentMethod")
dummy
data5=pd.concat([data5,dummy],axis=1)
data5
```

```
[22] #create dummies for the following columns:
     data5=pd.get_dummies(data5,columns=['MultipleLines'])
     data5=pd.get_dummies(data5,columns=['OnlineSecurity'])
     data5=pd.get_dummies(data5,columns=['OnlineBackup'])
     data5=pd.get_dummies(data5,columns=['DeviceProtection'])

     data5
```

Cont.

Dropped the CustomerID Column

```
#remove the column labeled "customerID"
data6=data5.drop(['customerID'], axis =1)
data6
```

Adjusted Types

```
#change "MonthlyCharges" to an integer type
data6['MonthlyCharges'] = data6.MonthlyCharges.astype(int)
```

```
#change "TotalCharges" to an integer type
data6['TotalCharges'] = data6.MonthlyCharges.astype(int)
```

New DTypes

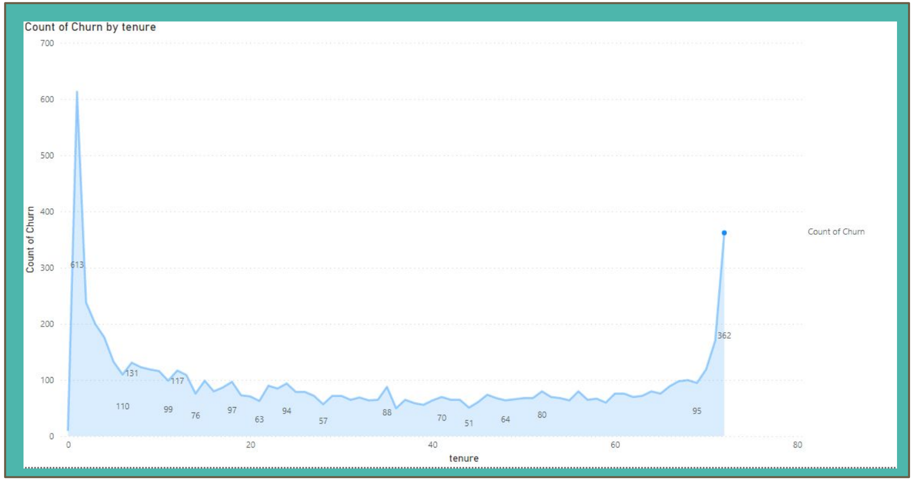
```
#list all datatypes within data6 dataset
data6.dtypes
```

```
SeniorCitizen                                int64
Partner                                      int64
Dependents                                   int64
tenure                                       int64
PhoneService                                 int64
TechSupport                                  int64
StreamingTV                                  int64
StreamingMovies                              int64
PaperlessBilling                             int64
MonthlyCharges                               int64
TotalCharges                                 int64
Churn                                        int64
Gender_Female                                uint8
Gender_Male                                  uint8
InternetService_DSL                          uint8
InternetService_Fiber optic                  uint8
InternetService_No                           uint8
Contract_Month-to-month                      uint8
Contract_One year                            uint8
Contract_Two year                            uint8
PaymentMethod_Bank transfer (automatic)      uint8
PaymentMethod_Credit card (automatic)        uint8
PaymentMethod_Electronic check               uint8
PaymentMethod_Mailed check                   uint8
MultipleLines_0                              uint8
MultipleLines_1                              uint8
OnlineSecurity_0                             uint8
OnlineSecurity_1                             uint8
OnlineBackup_0                               uint8
OnlineBackup_1                               uint8
OnlineBackup_No internet service             uint8
DeviceProtection_0                           uint8
DeviceProtection_1                           uint8
dtype: object
```

Exploratory Data Analysis (EDA)

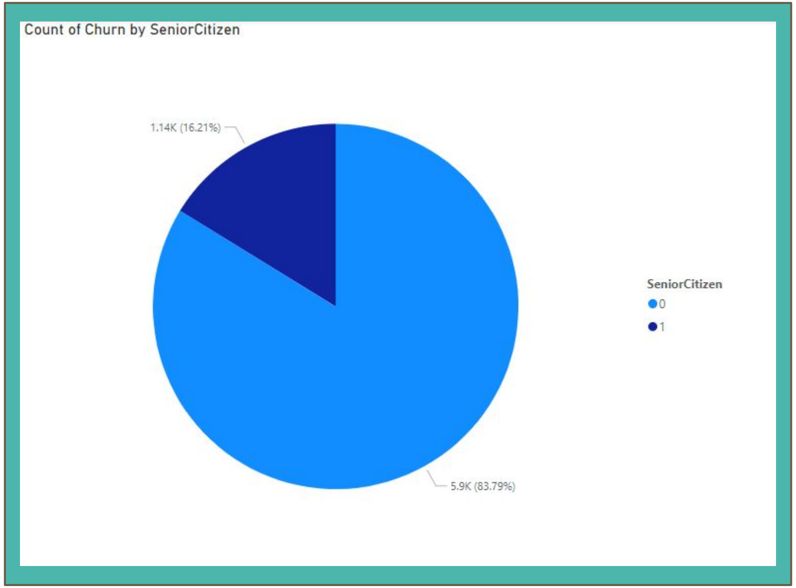
Purpose: Use Power BI for Data Visualization

EDA



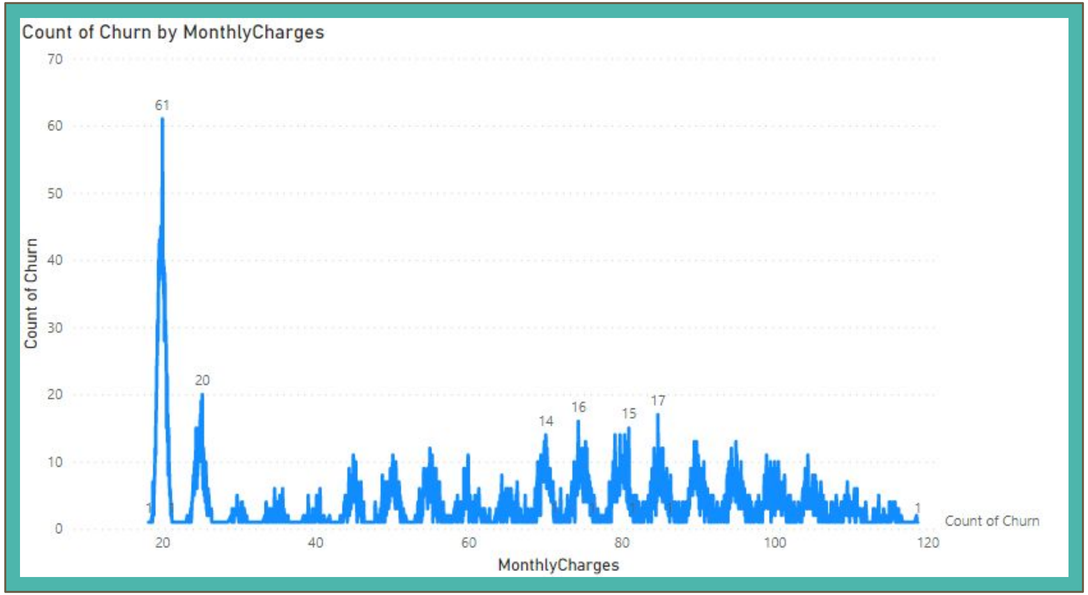
Tenure Analysis

Senior Citizen Analysis

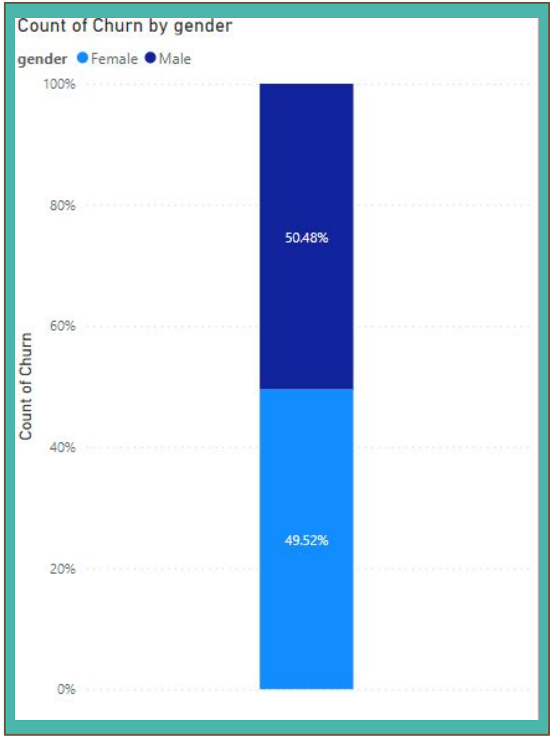


EDA

Gender Analysis



Monthly Charge Analysis



Splitting & Standardization of the Data

<u>Purpose:</u> Use Training & Validation Data Sets to Evaluate the Models

Splitting & Standardization of Data

Create Training & Test Data

```
#split data
#identify x as all columns except Churn
x = data6.loc[:,['SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling',
       'MonthlyCharges', 'TotalCharges', 'Gender_Female',
       'Gender_Male', 'InternetService_DSL', 'InternetService_Fiber optic',
       'InternetService_No', 'Contract_Month-to-month', 'Contract_One year',
       'Contract_Two year', 'PaymentMethod_Bank transfer (automatic)',
       'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check',
       'MultipleLines_0', 'MultipleLines_1', 'OnlineSecurity_0',
       'OnlineSecurity_1', 'OnlineBackup_0', 'OnlineBackup_1',
       'OnlineBackup_No internet service', 'DeviceProtection_0',
       'DeviceProtection_1']]
#identify y as "Churn" columns
y = data6.loc[:, ["Churn"]]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=0)
```

Standardization of Data

```
from sklearn.preprocessing import StandardScaler
standard_X=StandardScaler()
```

```
#transform the data for X_train to be reidentified
X_train = standard_X.fit_transform(X_train)
```

```
#transform data for X_test to be reidentified
X_test = standard_X.fit_transform(X_test)
```

Create X & Y

```
#Save the depedent feature "Churn_1" in a variable called "Y" and independent features in "X"
Y = data6["Churn"]
X = data6.drop("Churn",axis=1)
```

Technical Analysis Data Classification Models

Purpose: Test Various Models to Determine Best Option

3 Models

1. Support Vector Machines (SVM)
2. K-Nearest Neighbor (KNN)
3. Logistic Regression

Other Models: Linear Regression, Decision Trees, Random Forest

Support Vector Machines (SVM)

```
[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}

[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
```

SVM Code, accuracy score, best_params_

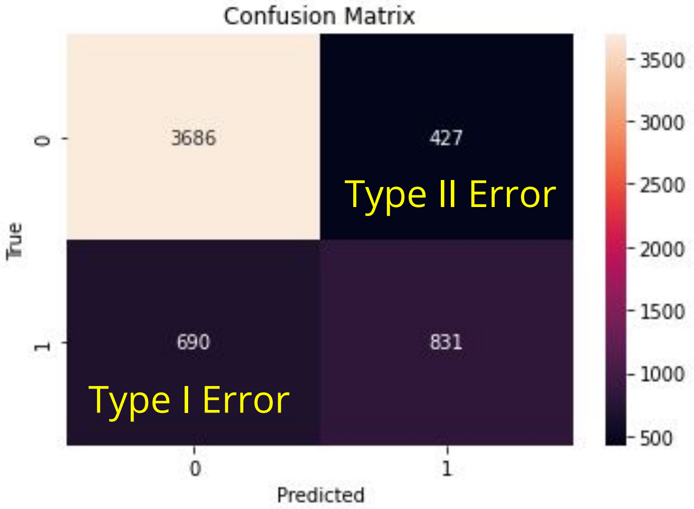
SVM: Confusion Matrix

Confusion Matrix

	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

SVM Plot

```
[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))
```



K-Nearest Neighbor (KNN)

KNN Train & Predict

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
#Train Model and Predict
k = 4
neigh = KNeighborsClassifier(n_neighbors = k).fit(xTrain,yTrain)
Pred_y = neigh.predict(xTest)
print("Accuracy of model at k = 4 is", metrics.accuracy_score(yTest, Pred_y))
```

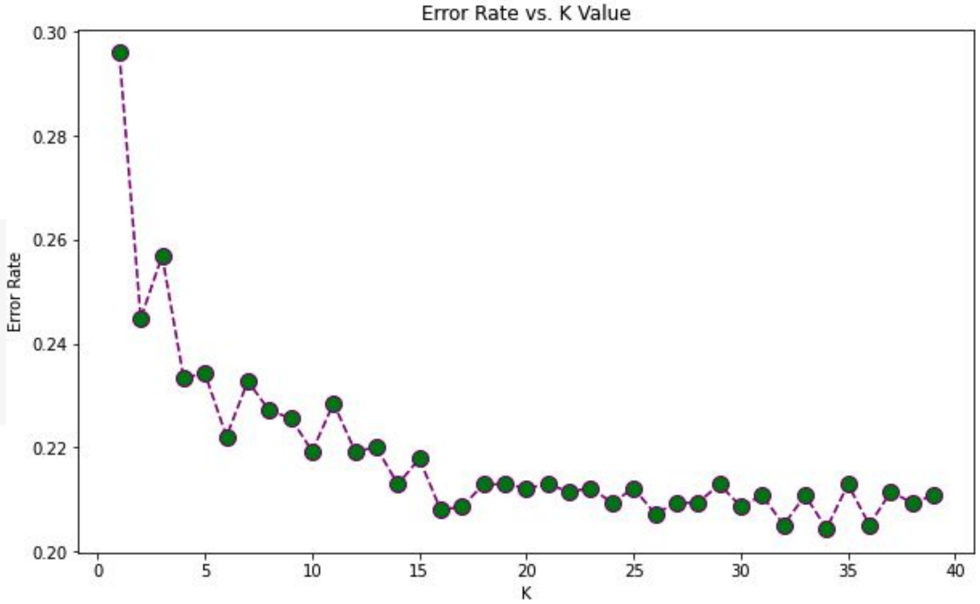
```
Accuracy of model at k = 4 is 0.7665010645848119
```

KNN Plot Error Rate

```
error_rate = []
for i in range (1,40):
  knn = KNeighborsClassifier(n_neighbors=i)
  knn.fit(xTrain, yTrain)
  pred_i = knn.predict(xTest)
  error_rate.append(np.mean(pred_i !=yTest))

plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='purple', linestyle='dashed', marker='o', markerfacecolor='green', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
print("Minimum error:-",min(error_rate),"at K =",error_rate.index(min(error_rate)))
```

```
Minimum error:- 0.2044002838892832 at K = 33
```



KNN Output Graph

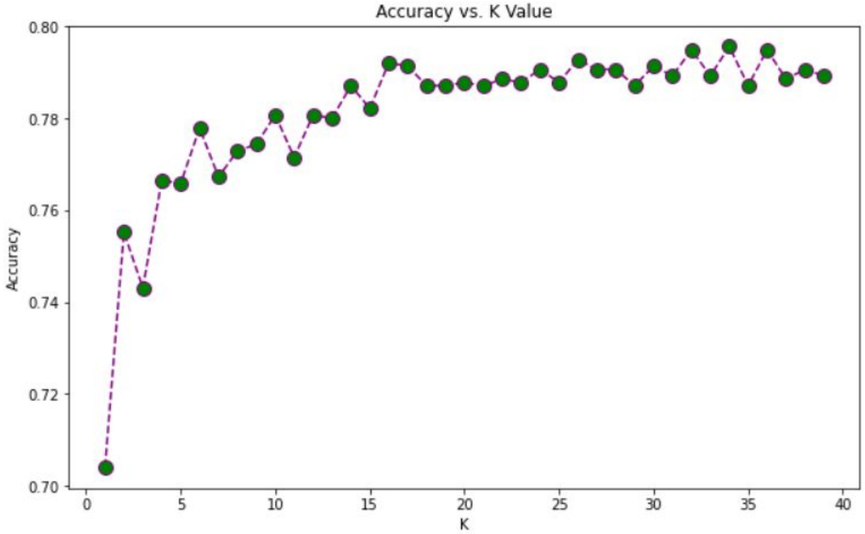
K-Nearest Neighbor (KNN)

```
[45] acc = []
     #Will take some time
     from sklearn import metrics
     for i in range (1,40):
       neigh = KNeighborsClassifier(n_neighbors = i).fit(xTrain,yTrain)
       yhat = neigh.predict(xTest)
       acc.append(metrics.accuracy_score(yTest,yhat))

     plt.figure(figsize=(10,6))
     plt.plot(range(1,40),acc,color='purple', linestyle='dashed', marker='o', markerfacecolor='green', markersize=10)
     plt.title('Accuracy vs. K Value')
     plt.xlabel('K')
     plt.ylabel('Accuracy')
     print("Maximum Accuracy:-",max(acc),"at K =",acc.index(max(acc)))
```

KNN Plot: Accuracy

```
Maximum Accuracy:- 0.7955997161107168 at K = 33
```



KNN Output: Accuracy vs. K Value

K-Nearest Neighbor (KNN)

```
#Scaling
scaler = StandardScaler()
scaler.fit(xTrain)
xTrain = scaler.transform(xTrain)
xTest = scaler.transform(xTest)
```

```
#Setting up to find the best parameters
n_neighbors = np.arange(1,40)
grid_params = {'n_neighbors' : n_neighbors,
              'leaf_size': [30, 35],
              'algorithm' : ['ball_tree', 'kd_tree']}
gridSearch = GridSearchCV(KNeighborsClassifier(), grid_params, verbose = 1, cv = 3, n_jobs = -1)
gridSearchresults = gridSearch.fit(xTrain, yTrain)
```

```
Fitting 3 folds for each of 156 candidates, totalling 468 fits
```

```
#finding the best parameters
gridSearchresults.best_params_
```

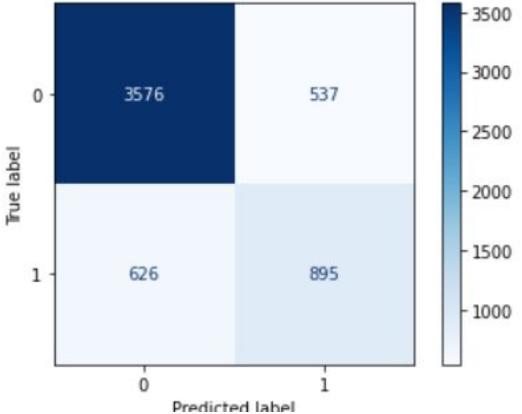
```
{'algorithm': 'ball_tree', 'leaf_size': 30, 'n_neighbors': 37}
```

```
#making confusion matrix
Classifier = KNeighborsClassifier(n_neighbors=37, leaf_size=30, algorithm ='ball_tree')
Classifier.fit(xTrain,yTrain)
y_pred_train= Classifier.predict(xTrain)
y_pred_test = Classifier.predict(xTest)
```

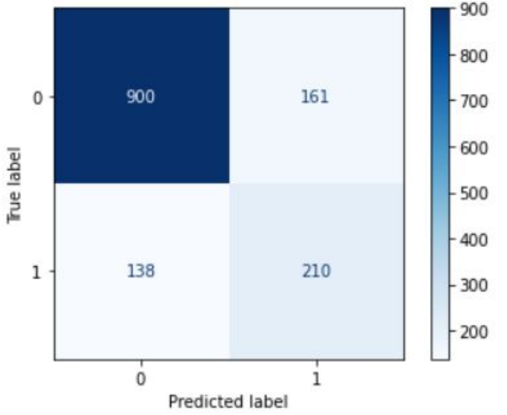
```
plot_confusion_matrix(Classifier,xTrain,yTrain, cmap= plt.cm.Blues)
```

```
plot_confusion_matrix(Classifier, xTest, yTest, cmap = plt.cm.Blues)
```

Training Data



Testing Data



Logistic Regression

Results: All Columns

```
                            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.
```

Results: P<0.05 Columns Only

```
                            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.
```

Logistic Regression

Accuracy Scores

```
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
```

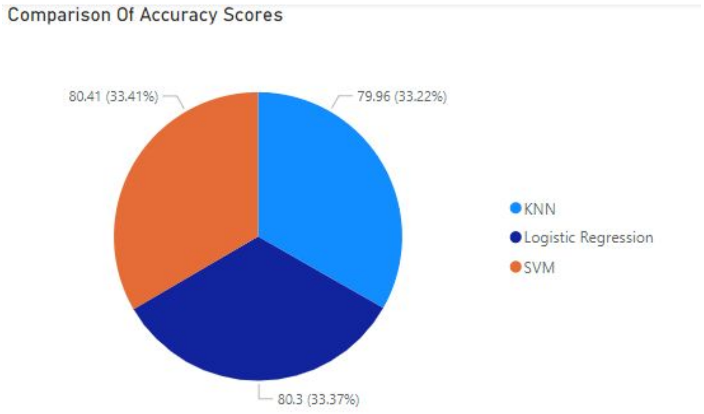
```
data6.corr()
```

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	TechSupport	StreamingTV	StreamingMovies	PaperlessBilling	Month
SeniorCitizen	1.000000	0.016479	-0.211185	0.016567	0.008576	-0.060625	0.105378	0.120176	0.156530	
Partner	0.016479	1.000000	0.452676	0.379697	0.017706	0.119999	0.124666	0.117412	-0.014877	
Dependents	-0.211185	0.452676	1.000000	0.159712	-0.001762	0.063268	-0.016558	-0.039741	-0.111377	
tenure	0.016567	0.379697	0.159712	1.000000	0.008448	0.324221	0.279756	0.286111	0.006152	
PhoneService	0.008576	0.017706	-0.001762	0.008448	1.000000	-0.096340	-0.022574	-0.032959	0.016505	
TechSupport	-0.060625	0.119999	0.063268	0.324221	-0.096340	1.000000	0.278070	0.279358	0.037880	
StreamingTV	0.105378	0.124666	-0.016558	0.279756	-0.022574	0.278070	1.000000	0.533094	0.223841	
StreamingMovies	0.120176	0.117412	-0.039741	0.286111	-0.032959	0.279358	0.533094	1.000000	0.211716	
PaperlessBilling	0.156530	-0.014877	-0.111377	0.006152	0.016505	0.037880	0.223841	0.211716	1.000000	
MonthlyCharges	0.220129	0.096913	-0.113910	0.247917	0.247277	0.338325	0.629562	0.627421	0.352138	
TotalCharges	0.220129	0.096913	-0.113910	0.247917	0.247277	0.338325	0.629562	0.627421	0.352138	
Churn	0.150889	-0.150448	-0.164221	-0.352229	0.011942	-0.164674	0.063228	0.061382	0.191825	
Gender_Female	0.001874	0.001808	-0.010517	-0.005106	0.006488	0.009212	0.008393	0.010487	0.011754	
Gender_Male	-0.001874	-0.001808	0.010517	0.005106	-0.006488	-0.009212	-0.008393	-0.010487	-0.011754	
InternetService_DSL	-0.108322	-0.000851	0.052010	0.013274	-0.452425	0.313118	0.016274	0.025698	-0.063121	
InternetService_Fiber optic	0.255338	0.000304	-0.165818	0.019720	0.289999	-0.020492	0.329349	0.322923	0.326853	
InternetService_No	-0.182742	0.000615	0.139812	-0.039062	0.172209	-0.336298	-0.415552	-0.418675	-0.321013	
Contract_Month-to-month	0.138360	-0.280865	-0.231720	-0.645561	-0.000742	-0.285241	-0.112282	-0.116633	0.169096	
Contract_One year	-0.046262	0.082783	0.068368	0.202570	-0.002791	0.095775	0.061612	0.064926	-0.051391	
Contract_Two year	-0.117000	0.248091	0.204613	0.558533	0.003519	0.240824	0.072049	0.073960	-0.147889	
PaymentMethod_Bank transfer (automatic)	-0.016159	0.110706	0.052021	0.243510	0.007556	0.101252	0.046252	0.048652	-0.016332	

Correlation for All Columns

Conclusion

- ***Recap:***
    - *Telco finding churn rate base on variables such as gender, age, average monthly charge, etc.*
- ***3 Classification Models:***
    - *Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Logistic Regression*
- ***BEST model:***
    - *Support Vector Machine (SVM)*



Thank You, Questions?