Customer Churn Prediction Using Machine Learning

Phase 4 Submission Document

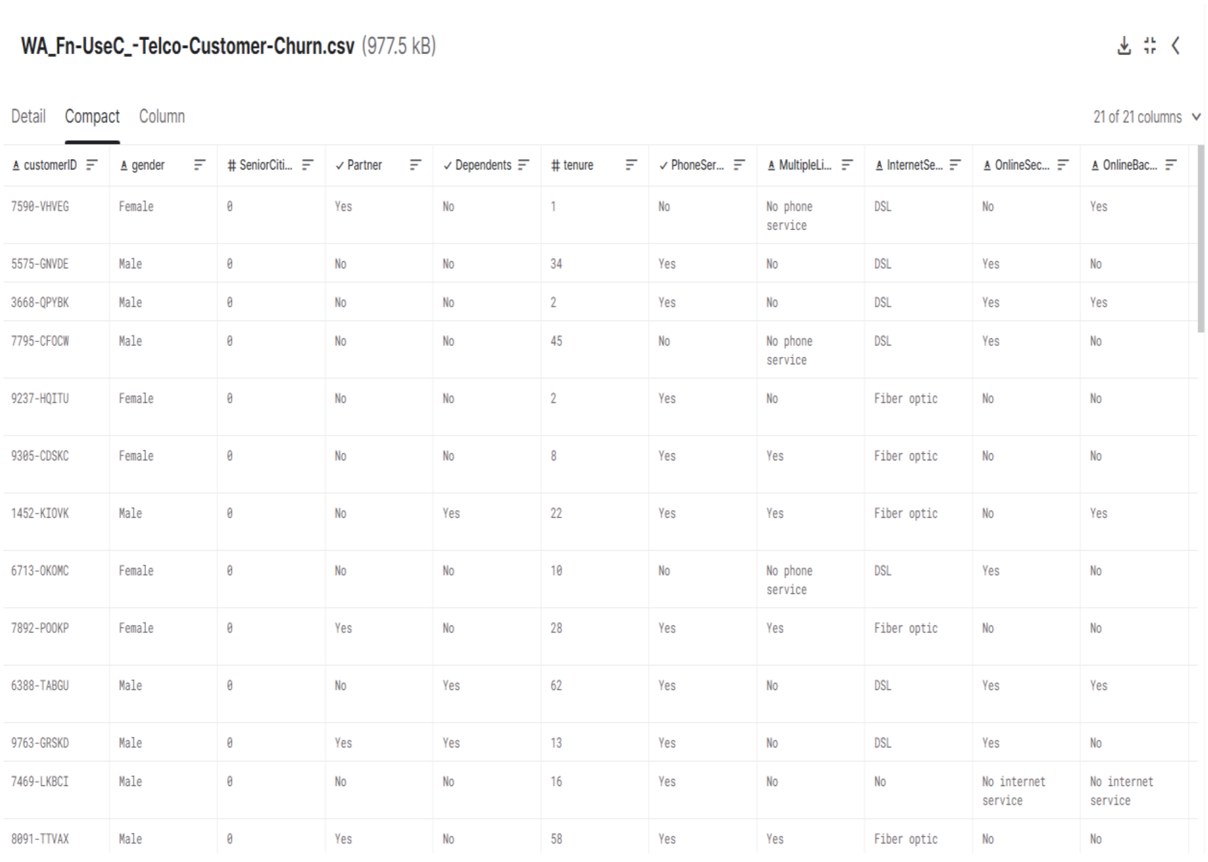
Name	J.S Thamizh Arasan
Reg. No	410121104057
NM ID	Au410121104057
Department	CSE-III
Domain	Data Analytics with Cognos
Project Title	Customer Churn Prediction
Phase 3	Development Part II
College	4101-Adhi College of Engineering and Technology, Kanchipuram

Customer Churn Prediction

Introduction to Telco Customer Churn:

In the dynamic and fiercely competitive telecommunications (telco) industry, customer churn is a persistent challenge that can significantly impact a company's bottom line and market position. Customer churn, also known as customer attrition or turnover, occurs when subscribers decide to switch their telecom service providers. This phenomenon is driven by a myriad of factors, including pricing, service quality, customer service, and evolving technology. To combat this issue, telco companies employ various strategies and initiatives aimed at retaining their customers, collectively known as customer retention programs. This introduction provides an overview of the critical concept of customer churn in the telco industry and the need for effective customer retention efforts to mitigate its negative consequences.

Given Data Set:



WA_Fn-UseC_-Telco-Customer-Churn.csv (977.5 kB)

Detail Compact Column

21 of 21 columns

customerID	gender	SeniorCiti...	Partner	Dependents	tenure	PhoneSer...	MultipleLi...	InternetSe...	OnlineSec...	OnlineBac...
7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No
3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No
9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No
9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No
1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes
6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	No
7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No
6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes	No	DSL	Yes	No
7469-LKBCI	Male	0	No	No	16	Yes	No	No	No internet service	No internet service
8091-TTVAX	Male	0	Yes	No	58	Yes	Yes	Fiber optic	No	No

Overview of the Process:

1.Data Collection:

- Gather historical customer data from various sources, including customer demographics, usage patterns, billing information, customer service interactions, and any other relevant information.
- Ensure that the dataset is comprehensive and includes information about both churned and non-churned customers.

2.Data Preprocessing:

- Clean and preprocess the data by handling missing values, dealing with outliers, and encoding categorical variables.
- Feature engineering: Create new features that can potentially improve the predictive power of the model. For example, you may calculate customer tenure or create a feature to represent the total number of customer service calls made.

3.Data Splitting:

- Divide the dataset into training and testing sets to evaluate the model's performance. A common split might be 70% for training and 30% for testing.

4.Feature Selection:

- Identify the most relevant features by using techniques like feature importance analysis or domain knowledge.
- Feature scaling: Normalize or standardize the data to ensure that all features have the same scale.

5.Model Selection:

- Choose a machine learning algorithm appropriate for the task. Common choices for customer churn prediction include logistic regression, decision trees, random forests, support vector machines, and neural networks.

6.Model Training:

- Train the selected model on the training data. The model learns to predict customer churn based on the historical data.

7.Model Evaluation:

- Assess the model's performance using metrics such as accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC).
- Use a confusion matrix to understand the true positives, true negatives, false positives, and false negatives.

8.Hyperparameter Tuning:

- Optimize the model's hyperparameters to improve its performance. You can use techniques like grid search or random search.

9.Model Validation:

- Validate the model's performance on the testing dataset to ensure that it generalizes well to unseen data.

10.Interpretability:

- Analyze the model's predictions and feature importance to understand which factors influence customer churn the most.

11.Deployment:

- Deploy the trained model to a production environment where it can make real-time predictions or generate insights for business decisions.

12.Monitoring:

- Continuously monitor the model's performance in the production environment and retrain it as needed with new data to adapt to changing customer behaviors.

13.Business Insights:

- Translate model predictions into actionable business insights to reduce churn. Identify strategies to retain at-risk customers and improve customer satisfaction.

PROCEDURE:

Data Collection: Gather historical customer data from various sources.

Data Preprocessing:_ Clean, handle missing values, and encode categorical variables.

Feature Engineering:_ Create relevant features to improve model performance.

Data Splitting:_ Divide the dataset into training and testing sets.

Model Selection:_Choose a machine learning algorithm (e.g., logistic regression, decision trees).

Model Training: Train the selected model on the training data.

Model Evaluation: Assess the model's performance with metrics like accuracy and F1 score.

Hyperparameter Tuning: Optimize the model's parameters for better results.

Model Validation: Validate the model's performance on the testing dataset.

Deployment: Deploy the trained model for real-time predictions.

Monitoring: Continuously monitor the model and retrain as needed.

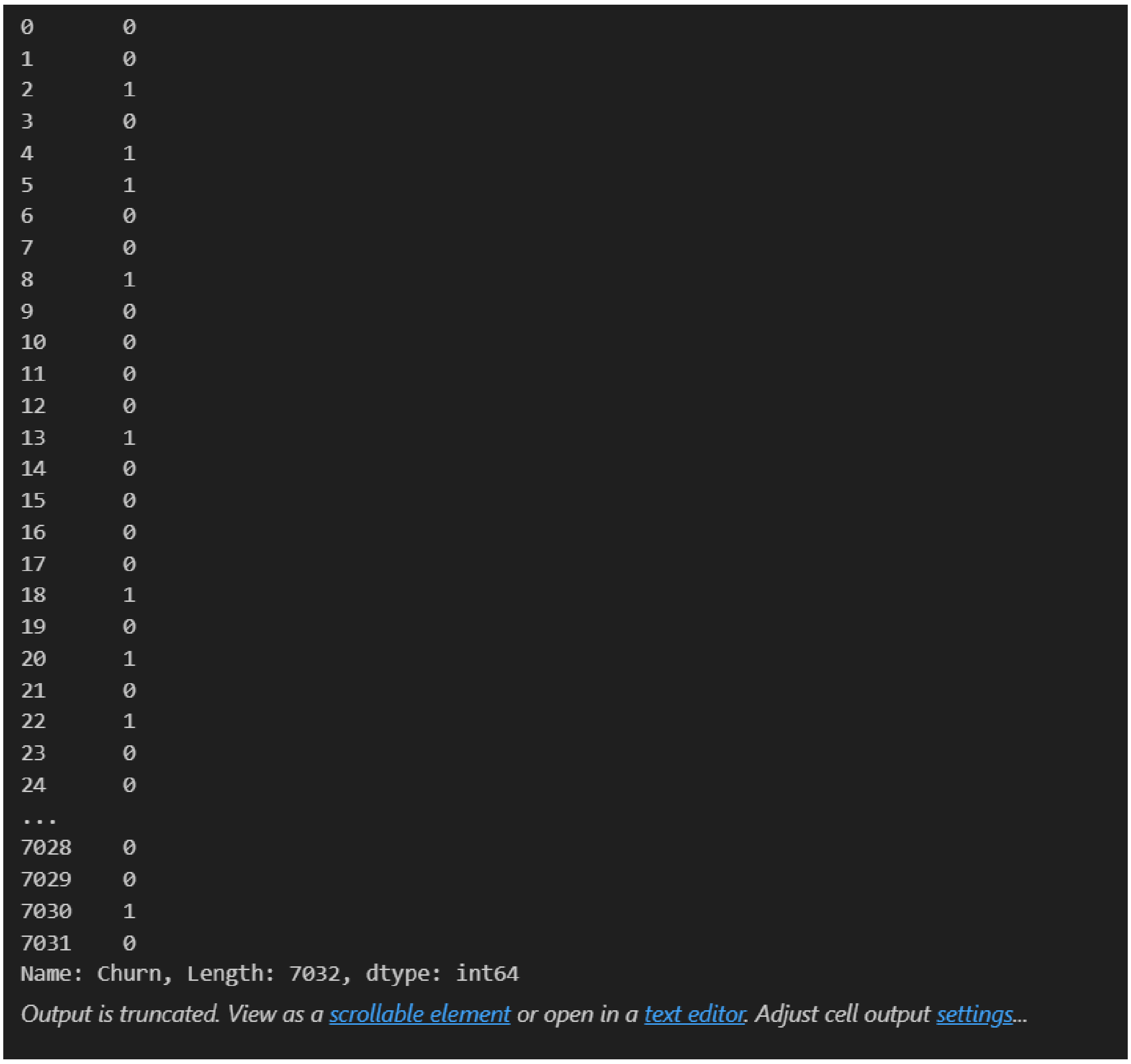
Actionable Insights: Translate model predictions into strategies to reduce churn.

Feature Selection:

Program:

```
df=df.drop('Unnamed: 0',axis=1)

x=df.drop('Churn',axis=1)

y=df['Churn']

y
```

Output:



```
0       0
1       0
2       1
3       0
4       1
5       1
6       0
7       0
8       1
9       0
10      0
11      0
12      0
13      1
14      0
15      0
16      0
17      0
18      1
19      0
20      1
21      0
22      1
23      0
24      0
...
7028    0
7029    0
7030    1
7031    0
Name: Churn, Length: 7032, dtype: int64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Checking for the Missing Data

Program:

1. Create a copy of base data for manupulation & processing

telco_data = telco_base_data.copy()

2. Total Charges should be numeric amount. Let's convert it to numerical data type

telco_data.TotalCharges=pd.to_numeric(telco_data.TotalCharges,errors='coerce’)

telco_data.isnull().sum()

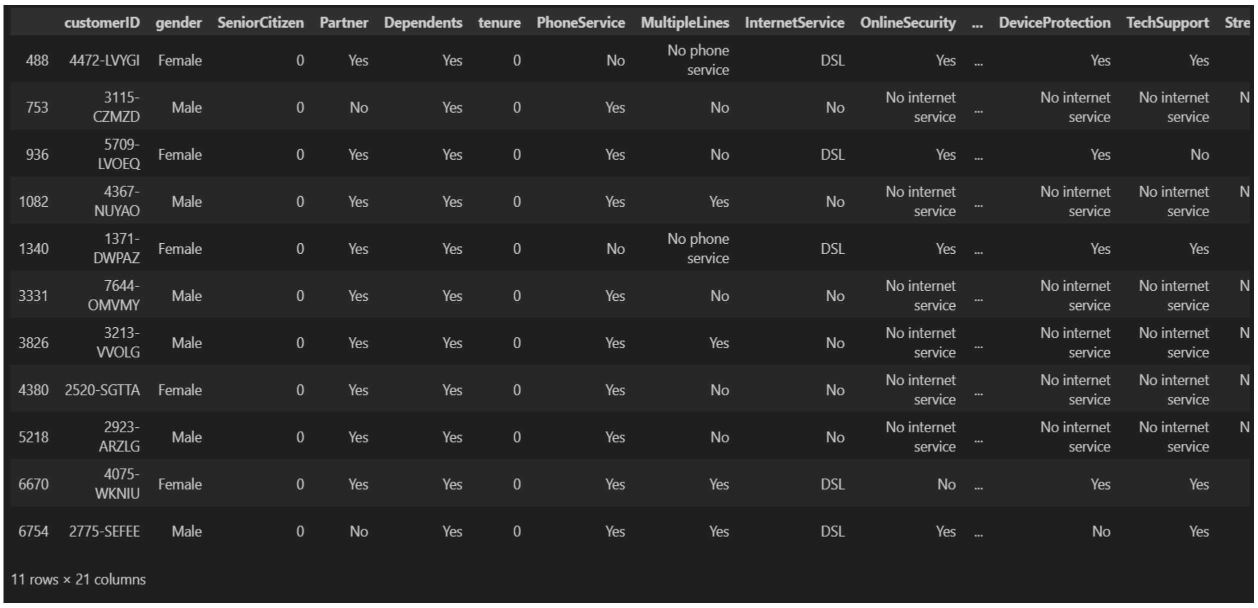
Output:

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

3. As we can see there are 11 missing values in TotalCharges column. Let's check these records

```
telco_data.loc[telco_data ['TotalCharges'].isnull() == True]
```

Output:



	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	Stre
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes	
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	N
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	...	Yes	No	
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service	N
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes	
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	N
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service	N
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	N
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	N
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	...	Yes	Yes	
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	...	No	Yes	

11 rows × 21 columns

4. Missing Value Treatement

Since the % of these records compared to total dataset is very low ie 0.15%, it is safe to ignore them from further processing.

```
#Removing missing values

telco_data.dropna(how = 'any', inplace = True)

#telco_data.fillna(0)
```

5. Divide customers into bins based on tenure e.g. for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24; so on...

```
# Get the max tenure

print(telco_data['tenure'].max()) #72
```

Output:

```
72
```

```
# Group the tenure in bins of 12 months

labels = ["{0}- {1}".format(i, i + 11) for i in range(1, 72, 12)]

telco_data['tenure_group'] = pd.cut(telco_data.tenure, range(1, 80, 12), right=False, labels=labels)

telco_data['tenure_group'].value_counts()
```

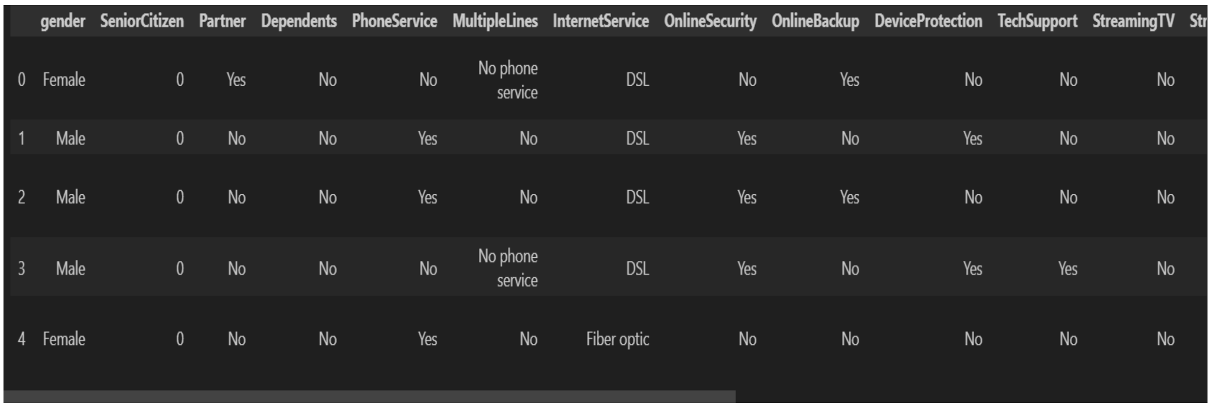
Output:

```
1- 12     2175
61- 72    1407
13- 24    1024
49- 60     832
25- 36     832
37- 48     762
Name: tenure_group, dtype: int64
```

6. Remove columns not required for processing

```
#drop column customerID and tenure
telco_data.drop(columns= ['customerID','tenure'], axis=1, inplace=True)
telco_data.head()
```

Output:



	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Str
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No	No	No	

Model Training:

Choose Machine learning Algorithm:

There are a number of different machine learning algorithms that can be used for Customer churn Prediction, such as Decision Tree, Random Forest and Performing PCA.

Machine Learning Models:

1.Decision Tree Classifier.

A decision tree classifier in data analytics is a machine learning algorithm that uses a tree-like structure to make decisions and classify data based on a set of rules or conditions. It recursively splits the data into subsets, using specific criteria at each branch, to ultimately predict the class or category to which a data point belongs. Decision trees are widely used for tasks like classification and regression analysis, offering a straightforward and interpretable way to make data-driven decisions.

Program:

```
model_dt=DecisionTreeClassifier(criterion="gini",random_state=100,max_depth
=6, min_samples_leaf=8)

model_dt.fit(x_train,y_train)
```

Output:

```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
y_pred=model_dt.predict(x_test)

y_pred
```

Output:

```
array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

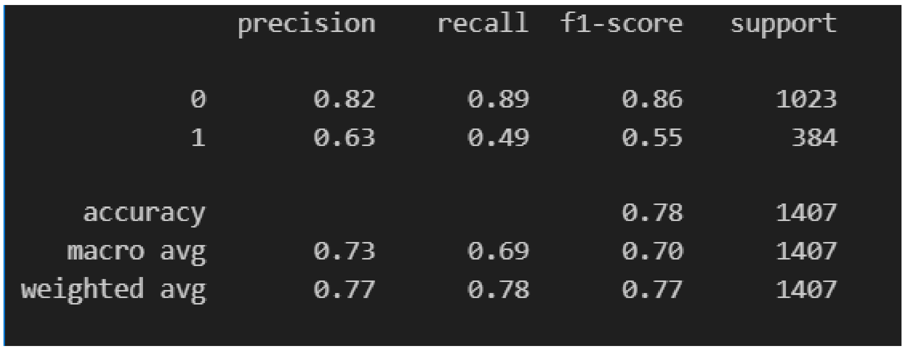
```
model_dt.score(x_test,y_test)
```

Output:

```
0.7818052594171997
```

print(classification_report(y_test, y_pred, labels=[0,1]))

Output:



```
              precision    recall  f1-score   support

           0       0.82      0.89      0.86      1023
           1       0.63      0.49      0.55       384

    accuracy                           0.78      1407
   macro avg       0.73      0.69      0.70      1407
weighted avg       0.77      0.78      0.77      1407
```

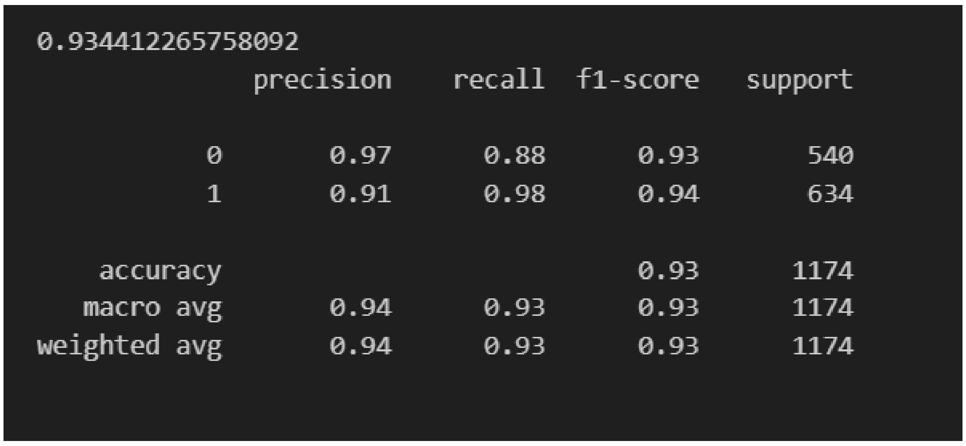
As you can see that the accuracy is quite low, and as it's an imbalanced dataset, we shouldn't consider Accuracy as our metrics to measure the model, as Accuracy is cursed in imbalanced datasets.

Hence, we need to check recall, precision & f1 score for the minority class, and it's quite evident that the precision, recall & f1 score is too low for Class 1, i.e. churned customers.

Hence, moving ahead to call SMOTEENN (UpSampling + ENN)

```
sm = SMOTEENN()

X_resampled, y_resampled = sm.fit_sample(x,y)

xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_size=0.2)

model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6,
min_samples_leaf=8)

model_dt_smote.fit(xr_train,yr_train)

yr_predict = model_dt_smote.predict(xr_test)

model_score_r = model_dt_smote.score(xr_test, yr_test)

print(model_score_r)

print(metrics.classification_report(yr_test, yr_predict))
```

Output:



```
0.934412265758092
              precision    recall  f1-score   support

           0       0.97      0.88      0.93       540
           1       0.91      0.98      0.94       634

    accuracy                           0.93      1174
   macro avg       0.94      0.93      0.93      1174
weighted avg       0.94      0.93      0.93      1174
```

print(metrics.confusion_matrix(yr_test, yr_predict))

Output:

```
[[477  63]
 [ 14 620]]
```

Now we can see quite better results, i.e. Accuracy: 92 %, and a very good recall, precision & f1 score for minority class.

Let's try with some other classifier.

2. Random Forest Classifier:

A random forest classifier in data analytics is an ensemble machine learning algorithm that combines multiple decision trees to improve the accuracy and reliability of classification tasks. It works by generating a collection of decision trees and then aggregating their predictions to make a final classification. This ensemble approach helps reduce overfitting and enhances the overall performance of the model, making it a powerful tool for various classification problems.

Program:

```
from sklearn.ensemble import RandomForestClassifier

model_rf=RandomForestClassifier(n_estimators=100,criterion='gini',random_state=100,max_depth=6
, min_samples_leaf=8)

model_rf.fit(x_train,y_train)
```

Output:

```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

y_pred=model_rf.predict(x_test)

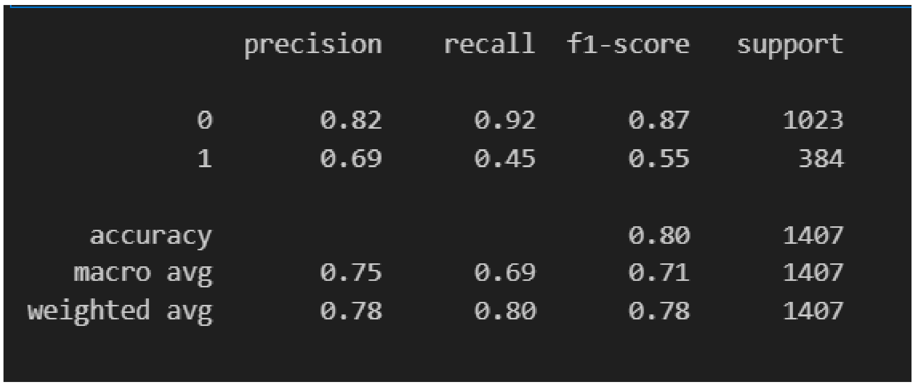
model_rf.score(x_test,y_test)

Output:

0.7953091684434968

print(classification_report(y_test, y_pred, labels=[0,1]))

Output:



```
              precision    recall  f1-score   support

           0       0.82      0.92      0.87      1023
           1       0.69      0.45      0.55       384

    accuracy                           0.80      1407
   macro avg       0.75      0.69      0.71      1407
weighted avg       0.78      0.80      0.78      1407
```

sm = SMOTEENN()

X_resampled1, y_resampled1 = sm.fit_sample(x,y)

xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(X_resampled1, y_resampled1,test_size=0.2)

model_rf_smote=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)

model_rf_smote.fit(xr_train1,yr_train1)

Output:

RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)

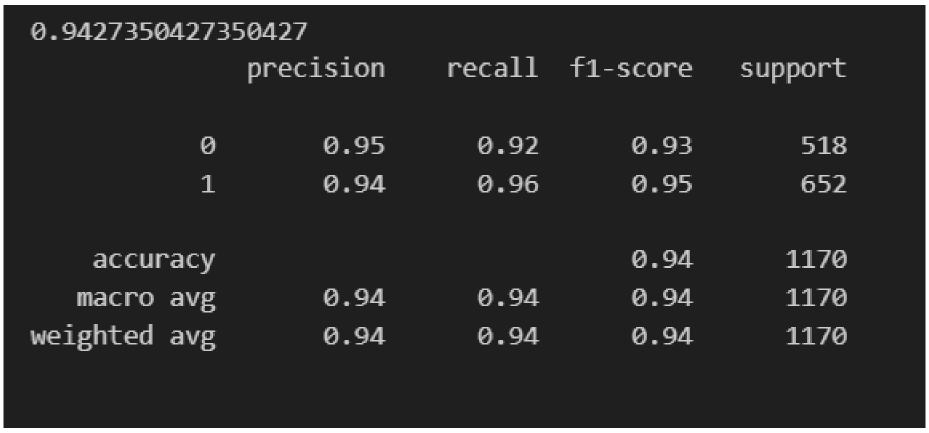
yr_predict1 = model_rf_smote.predict(xr_test1)

model_score_r1 = model_rf_smote.score(xr_test1, yr_test1)

print(model_score_r1)

print(metrics.classification_report(yr_test1, yr_predict1))

Output:



```
0.9427350427350427
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       518
           1       0.94      0.96      0.95       652

    accuracy                           0.94      1170
   macro avg       0.94      0.94      0.94      1170
weighted avg       0.94      0.94      0.94      1170
```

print(metrics.confusion_matrix(yr_test1, yr_predict1))

Output:

```
[[478  40]
 [ 27 625]]
```

With RF Classifier, also we are able to get quite good results, infact better than Decision Tree.

3.Performing PCA:

```
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(0.9)
xr_train_pca = pca.fit_transform(xr_train1)
xr_test_pca = pca.transform(xr_test1)
explained_variance = pca.explained_variance_ratio_
model=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 
100,max_depth=6, min_samples_leaf=8)
model.fit(xr_train_pca,yr_train1)
```

Output:

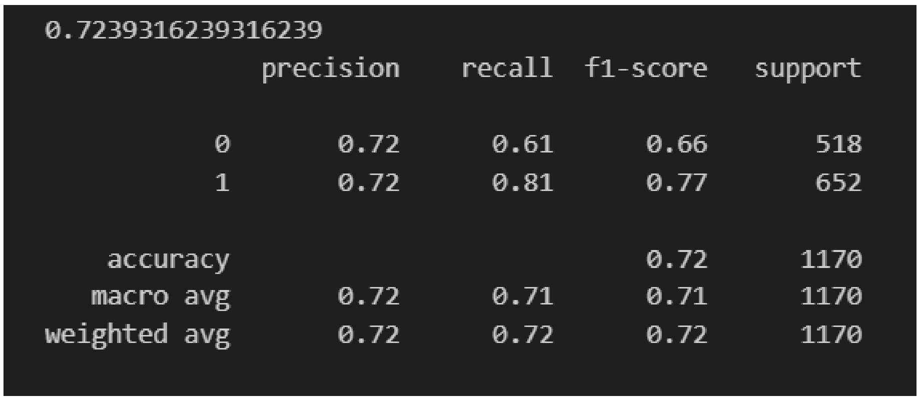
```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
yr_predict_pca = model.predict(xr_test_pca)
model_score_r_pca = model.score(xr_test_pca, yr_test1)
```

print(model_score_r_pca)

print(metrics.classification_report(yr_test1, yr_predict_pca))

Output:



```
0.7239316239316239
              precision    recall  f1-score   support

           0       0.72      0.61      0.66       518
           1       0.72      0.81      0.77       652

    accuracy                           0.72      1170
   macro avg       0.72      0.71      0.71      1170
weighted avg       0.72      0.72      0.72      1170
```

Model Training:

Data Preprocessing:

- ➢ Clean the dataset: Handle missing values and outliers.
- ➢ Encode categorical variables: Convert categorical features into numerical format using techniques like one-hot encoding or label encoding.
- ➢ Feature scaling: Normalize or standardize numerical features to ensure they are on the same scale.
- ➢ Feature selection: Identify and select relevant features that have the most impact on churn prediction.

Data Splitting:

- ➢ Divide the dataset into two parts: a training set and a testing set. A common split is 70% for training and 30% for testing.

Model Selection:

- ➢ Choose an appropriate machine learning algorithm. Common choices for customer churn prediction include logistic regression, decision trees, random forests, support vector machines, and neural networks.

Model Training:

- ➢ Train the selected model on the training dataset. The model learns to make predictions based on historical data.
- ➢ The training process involves optimizing the model's internal parameters to minimize prediction errors.

Model Evaluation:

- Assess the model's performance using relevant evaluation metrics. Common metrics include:
- Accuracy: The proportion of correct predictions.
- Precision: The ability of the model to correctly predict positive cases.
- Recall: The ability of the model to identify all positive cases.
- F1 Score: The harmonic mean of precision and recall.
- ROC-AUC: The area under the receiver operating characteristic curve.
- Hyperparameter Tuning (Optional):
- Fine-tune the model by adjusting hyperparameters (e.g., learning rate, regularization strength, tree depth) to optimize performance.

Model Validation:

- Validate the model's performance on the testing dataset to ensure it generalizes well to unseen data. This step helps identify if the model is overfitting or underfitting.
- Model Deployment (in the production environment):
- If the model meets the desired performance criteria, deploy it to a production environment where it can make real-time predictions or provide insights.

Monitoring:

- Continuously monitor the model's performance in the production environment to ensure it remains accurate and up-to-date. Retrain the model as needed with new data.

Split the data into training and test sets:

```
#Removing missing values

telco_data.dropna(how = 'any', inplace = True)

#telco_data.fillna(0)
```

Divide customers into bins based on tenure e.g. for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24; so on...

```
# Get the max tenure

print(telco_data['tenure'].max()) #72
```

Output:

```
72
```

```
# Group the tenure in bins of 12 months

labels = ["{0}- {1}".format(i, i + 11) for i in range(1, 72, 12)]

telco_data['tenure_group'] = pd.cut(telco_data.tenure, range(1, 80, 12), right=False, labels=labels)

telco_data['tenure_group'].value_counts()
```

Output:

```
1- 12    2175

61- 72   1407

13- 24   1024

49- 60    832

25- 36    832

37- 48    762

Name: tenure_group, dtype: int64
```

Train the model on the training set: This involves feeding the training data to the model ad allowing it to learn the relationship between the features an the target variable.

Model evaluation:

Metrics: Evaluate the model's performance using key metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.

Confusion Matrix: Analyze the confusion matrix to understand true positives, true negatives, false positives, and false negatives.

Validation: Validate the model's performance on a separate testing dataset to ensure it generalizes well to unseen data.

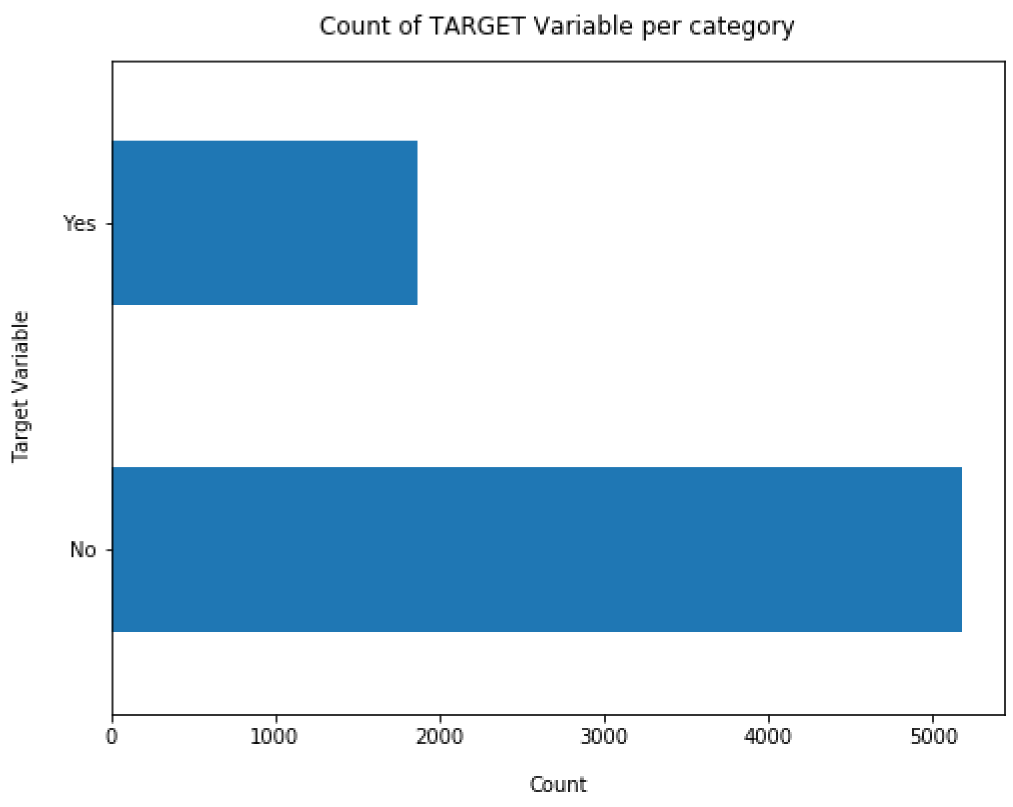
Fine-Tuning: Optionally, fine-tune hyperparameters to optimize model performance.

Continuous Monitoring: Continuously monitor the model in a production environment and retrain as needed to adapt to changing customer behaviors.

Evaluation of Predicted Data:

```
telco_base_data['Churn'].value_counts().plot(kind='barh', figsize=(8, 6))

plt.xlabel("Count", labelpad=14)

plt.ylabel("Target Variable", labelpad=14)

plt.title("Count of TARGET Variable per category", y=1.02);
```

Output:



```
100*telco_base_data['Churn'].value_counts()/len(telco_base_data['Churn'])
```

Output:

```
No     73.463013

Yes    26.536987

Name: Churn, dtype: float64
```

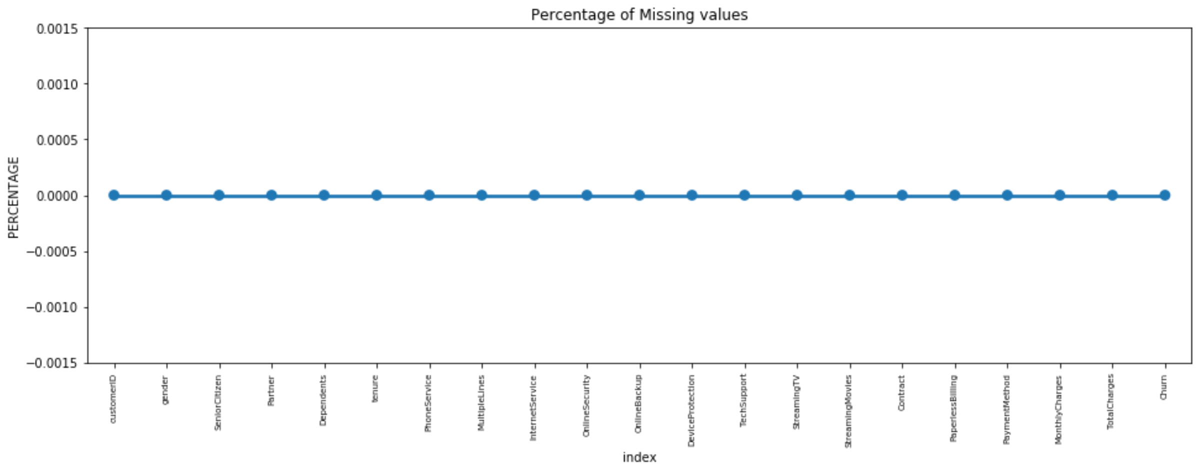
```
telco_base_data['Churn'].value_counts()
```

Output:

```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

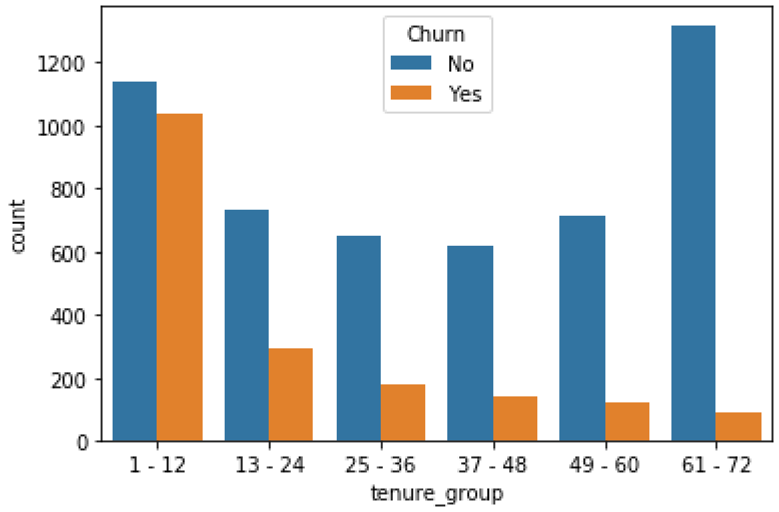
```
missing = pd.DataFrame((telco_base_data.isnull().sum())*100/telco_base_data.shape[0]).reset_index()
plt.figure(figsize=(16,5))
ax = sns.pointplot('index',0,data=missing)
plt.xticks(rotation =90,fontsize =7)
plt.title("Percentage of Missing values")
plt.ylabel("PERCENTAGE")
plt.show()
```

Output:



```
for i, predictor in enumerate(telco_data.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
    plt.figure(i)
    sns.countplot(data=telco_data, x=predictor, hue='Churn')
```

Output:

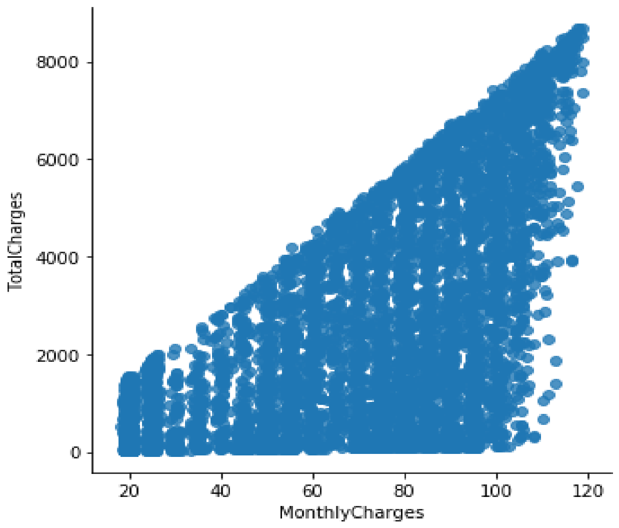


Relationship between Monthly Charges and Total Charges

```
sns.lmplot(data=telco_data_dummies, x='MonthlyCharges', y='TotalCharges', fit_reg=False)
```

Output:

```
<seaborn.axisgrid.FacetGrid at 0x20d8a9289e8>
```



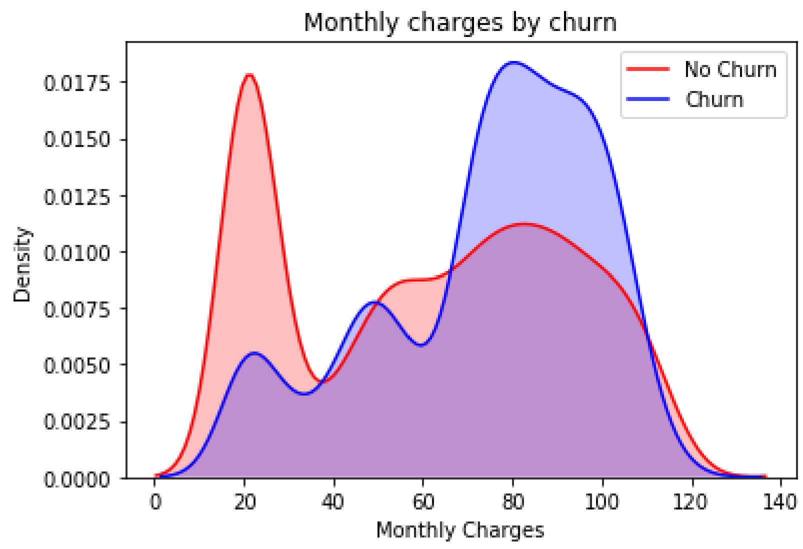
Total Charges increase as Monthly Charges increase - as expected.

Churn by Monthly Charges and Total Charges

```
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 0) ],
            color="Red", shade = True)
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 1) ],
            ax =Mth, color="Blue", shade= True)
Mth.legend(["No Churn","Churn"],loc='upper right')
Mth.set_ylabel('Density')

Mth.set_xlabel('Monthly Charges')

Mth.set_title('Monthly charges by churn')
```

Output:

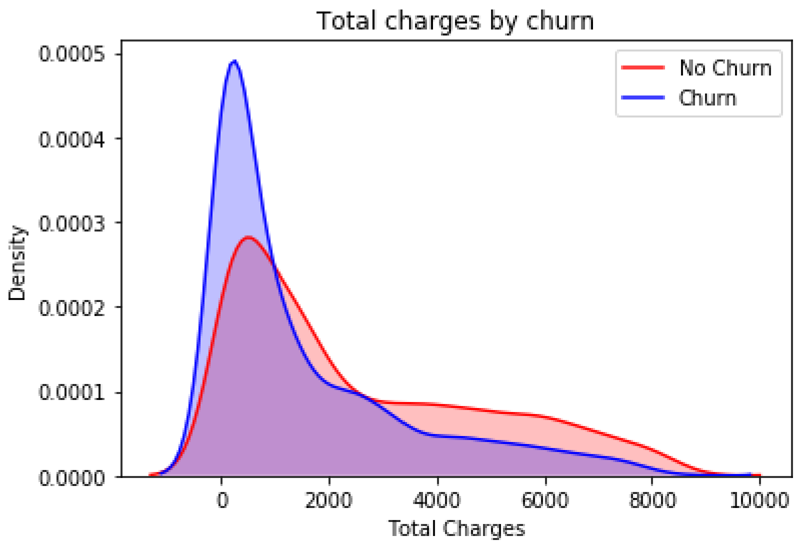
```
Text(0.5, 1.0, 'Monthly charges by churn')
```



Insight: Churn is high when Monthly Charges ar high

```
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 0) ],
            color="Red", shade = True)

Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 1) ],
            ax =Tot, color="Blue", shade= True)

Tot.legend(["No Churn","Churn"],loc='upper right')

Tot.set_ylabel('Density')

Tot.set_xlabel('Total Charges')

Tot.set_title('Total charges by churn')
```

Output:



**Surprising insight ** as higher Churn at lower Total Charges

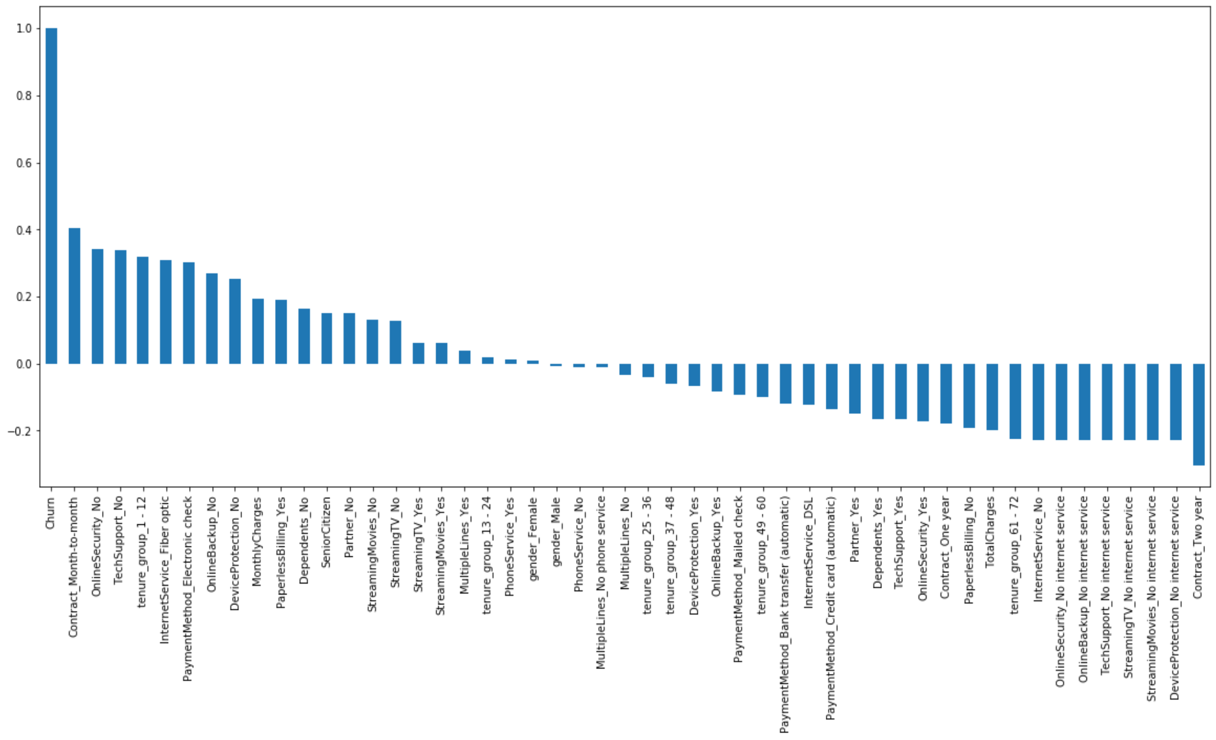
Build a corelation of all predictors with 'Churn'

```
plt.figure(figsize=(20,8))
```

```
telco_data_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20d8a979f98>
```



Pickling the model:

```
import pickle

filename = 'model.sav'

pickle.dump(model_rf_smote, open(filename, 'wb'))

load_model = pickle.load(open(filename, 'rb'))

model_score_r1 = load_model.score(xr_test1, yr_test1)

model_score_r1
```

Output:

```
0.9427350427350427
```

Our final model i.e. RF Classifier with SMOTEENN, is now ready and dumped in model.sav, which we will use and prepare API's so that we can access our model from UI.

Feature Engineering:

Tenure: Calculate the customer's tenure with the company, which represents the length of the customer relationship.

Customer Interaction: Create features like the number of customer service calls, complaints, or interactions to capture customer engagement.

Usage Patterns: Develop features that describe usage behavior, such as the total minutes of phone usage, data consumption, or number of text messages.

Billing Information: Extract relevant billing details like monthly charges, total charges over time, or payment history.

Contract Type: Encode the contract type (e.g., month-to-month, one-year, two-year) as a numerical feature.

Demographics: Utilize customer demographics like age, gender, and location to understand how they relate to churn.

Customer Feedback: If available, incorporate sentiment analysis of customer feedback or surveys as features.

Churn History: Create a binary feature indicating whether the customer has churned in the past.

Various features of perform model training:

In a Telco customer churn prediction project during the model training process, various features can be considered to enhance the effectiveness of the predictive model. These features include:

Customer Demographics:

- Age
- Gender
- Location
- Marital status

Customer Account Information:

- Contract type (e.g., month-to-month, one-year, two-year)
- Monthly charges
- Total charges over time
- Payment history

Customer Behavior and Usage:

- Tenure (length of customer relationship)
- Total minutes of phone usage
- Data consumption

- Number of text messages
- Number of customer service calls
- Complaint history

<u>Churn History:</u>

Binary feature indicating if the customer has churned in the past

<u>Billing Information:</u>

- Billing method (e.g., electronic check, credit card)
- Billing frequency
- Auto-pay status

<u>Customer Feedback and Satisfaction:</u>

- Results of customer satisfaction surveys
- Sentiment analysis of customer feedback

<u>Product or Service Usage:</u>

- Specific services used (e.g., TV, internet, phone)
- Service add-ons or packages

<u>Customer Loyalty Programs:</u>

Membership in loyalty programs or discounts

<u>Competitive Market Data:</u>

Competitive pricing and offers in the market

<u>Financial Indicators:</u>

Customer's credit score or financial stability

<u>Social Media Activity:</u>

Analyzing the customer's social media activity for sentiment or engagement

<u>Time-Dependent Features:</u>

Changes in customer behavior and patterns over time

Conclusion:

These are some of the quick insights from this exercise:

1. Electronic check medium are the highest churners.

2. Contract Type - Monthly customers are more likely to churn because of no contract terms, as they are free to go customers.

3. No Online security, No Tech Support category are high churners.

4. Non senior Citizens are high churners.

*****