विश्वजीवनामृतं ज्ञानम्

ABV-Indian Institute of Information Technology & Management, Gwalior

Customer Churn Prediction

of

Business Analytics

Mithil Jogi	2021IMG-033
Muthiah Sivavelan	2021IMG-034

Submitted To*:* Prof. Amandeep Kaur

Table of Contents

Chapters:

Introduction

In a competitive business environment, customer retention is critical to long-term success. Customer churn can hurt a company's profitability. This report describes a machine learning system that can predict customer trends, use historical data to identify high-risk customers, and provide organizations with retention strategies. We will provide an overview of our approach, data, design, and performance evaluation. This review is designed to provide companies in the industry or business with insights to reduce customer churn, ultimately increasing profitability, customer satisfaction and job completion.

Certainly! Here's an example of an implementation workflow for a customer churn prediction project:

Implementation Workflow for Customer Churn Prediction Project

1. <u>Data Collection:</u>

- Gather historical customer data, including demographic information, transaction records, customer interactions, and any other relevant features.
- Ensure data quality by addressing missing values, outliers, and inconsistencies.

2. <u>Data Preprocessing:</u>

- Perform data cleaning, including handling missing values and outliers.
- Feature engineering: Create new features or transformations that might improve model performance, such as customer tenure, frequency of interactions, and aggregate metrics.
- Data scaling or normalization if required.

3. Exploratory Data Analysis (EDA):

- Conduct EDA to gain insights into the dataset.
- Create visualizations to understand the distribution of key variables, identify trends, and detect patterns related to churn.
- Analyze correlations between features and churn.

4. Data Splitting:

- Split the dataset into a training set and a testing/validation set. Common splits are 70-30, 80-20, or 90-10, depending on the dataset size.

5. Model Selection:

- Choose machine learning algorithms suitable for classification tasks like logistic regression, decision trees, random forests, support vector machines, or gradient boosting.
- Consider employing ensemble techniques to combine multiple models for improved predictive performance.

6. Model Development:

- Train the selected models on the training dataset.
- Tune hyperparameters using techniques like cross-validation to optimize model performance.
- Validate the models using the testing/validation set.

7. Model Evaluation:

- Assess the performance of the models using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score, ROC AUC).
- Compare models to identify the best-performing one.

8. Feature Importance Analysis:

- Determine which features have the most influence on churn prediction.
- Use techniques such as feature importance scores from tree-based models or permutation feature importance.

9. Model Interpretation:

- Interpret model predictions to understand why certain customers are more likely to churn.
- Identify actionable insights that can guide business strategies.

Objective

The objective of a customer churn prediction project is to develop a predictive model that can identify customers who are likely to stop using a company's products or services in the near future. The primary goals of such a project typically include:

1. Proactive Customer Retention:
The project aims to help businesses identify and target customers at risk of churning before they actually do. This allows the company to take proactive steps to retain these customers and maintain their loyalty.

2. Cost Reduction:
Churn can be expensive for businesses, as acquiring new customers is often more costly than retaining existing ones. By predicting churn, companies can reduce the costs associated with acquiring replacement customers.

3. Improved Customer Satisfaction:
Identifying and addressing the causes of churn can lead to improvements in customer satisfaction and the overall customer experience. Satisfied customers are more likely to remain loyal.

4. Optimized Resource Allocation:
By knowing which customers are at risk, businesses can allocate resources, such as marketing efforts and special offers, more efficiently, focusing on those who are most likely to benefit from such actions.

5. Data-Driven Decision-Making: Churn prediction projects provide valuable insights into the factors and behaviors that lead to churn. These insights can inform strategic decisions and help shape business practices.

In summary, the primary objective of a customer churn prediction project is to leverage data and machine learning to reduce customer churn, enhance customer retention, and ultimately improve the financial health and competitiveness of a business.

Problem Statement

Certainly, here's a concise problem statement for a customer churn prediction project in five points:

1. Churn Impact:
Customer churn is impacting our business's profitability, as we are losing valuable customers who cease using our products or services.

2. Resource Inefficiency:
Our current customer retention strategies lack precision, leading to inefficient allocation of resources. We need a more data-driven approach to target customers at risk.

3. Lack of Predictive Insights:
The existing system does not provide predictive insights into which customers are likely to churn. We often react to churn after it has occurred, missing opportunities for proactive intervention.

4. Customer Satisfaction:
Churn not only affects revenue but also reflects negatively on customer satisfaction. Addressing churn is essential to maintain a positive customer experience.

5. Competitive Advantage:
Competing in the market requires us to stay ahead in customer retention. Developing a customer churn prediction model will give us a competitive edge by allowing us to act strategically and retain more customers.

Data Sourcing

Link of dataset used:
https://www.kaggle.com/datasets/mnassrib/telecom-churn-datasets

Using a dataset from Kaggle is a common and practical approach for data sourcing in a machine learning project.
Data Sourcing from Kaggle:

1. Selecting the Dataset:
Kaggle is a popular platform for hosting and sharing datasets related to various domains. When choosing a dataset for your customer churn prediction project, consider datasets that are relevant to your industry or business type. It's essential to select a dataset that aligns with your project's objectives and the type of customer information you need to predict churn accurately.

2. Data Licensing:
Ensure that you review the licensing terms and conditions associated with the Kaggle dataset you choose. Some datasets may have specific licensing restrictions, and it's important to comply with these terms, especially if you plan to use the model for commercial purposes.

3. Data Exploration:
Before diving into model development, thoroughly explore the dataset you've obtained from Kaggle. Understand the structure of the data, the meaning of each variable, and any potential data quality issues. Pay attention to missing values, outliers, and any data preprocessing steps required.

4. Data Attribution:
When using a dataset from Kaggle in your project, it's good practice to provide proper attribution to the dataset's source and creators. This ensures transparency and acknowledges the contributors who made the data available.

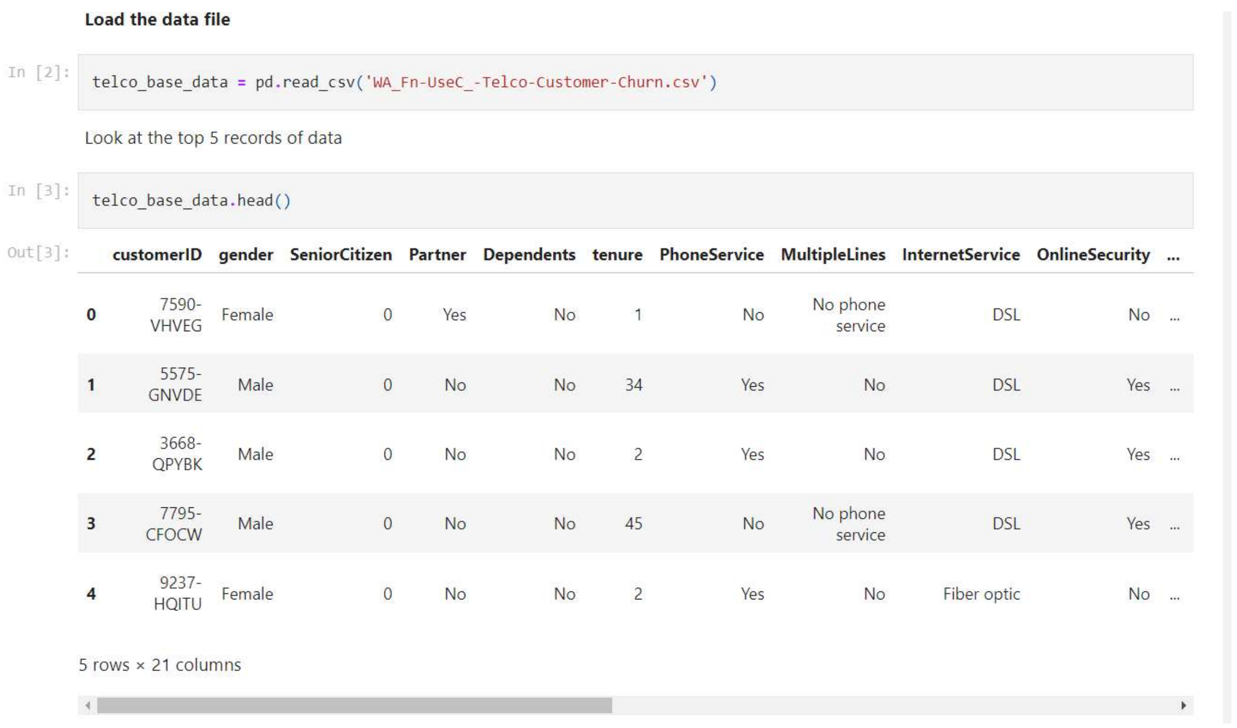
5. <u>Data Updates:</u>
Depending on the dataset's nature, you may want to periodically check for updates or newer versions. Some datasets on Kaggle may receive updates from the community, which can include additional data or improvements to existing data.

6. <u>Data Privacy and Compliance:</u>
If the dataset contains sensitive or personal information, be sure to handle it in compliance with data privacy regulations like GDPR, HIPAA, or other applicable laws, and take appropriate steps to protect customer privacy.

7. <u>Data Documentation:</u>
Kaggle datasets often come with descriptions, documentation, or discussions that can be valuable for understanding the dataset and its context. Utilize this information to gain insights into the data's characteristics and any potential issues.

Exploratory Data Analysis

Information about Dataset



Load the data file

```
In [2]: telco_base_data = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

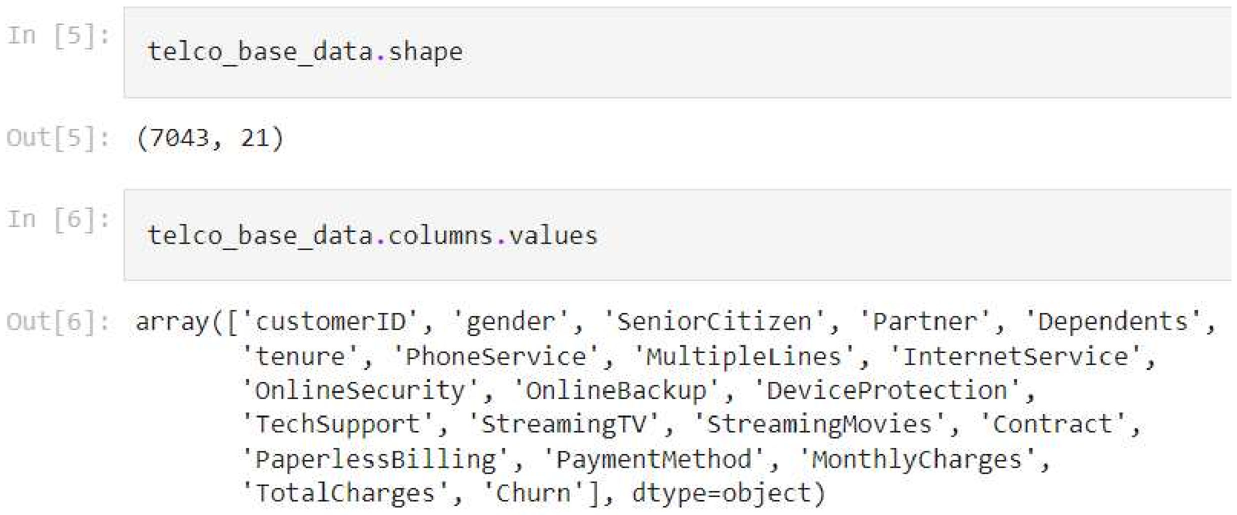
Look at the top 5 records of data

```
In [3]: telco_base_data.head()
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...

5 rows × 21 columns



```
In [5]: telco_base_data.shape

Out[5]: (7043, 21)

In [6]: telco_base_data.columns.values

Out[6]: array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
               'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
               'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
               'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
               'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
               'TotalCharges', 'Churn'], dtype=object)
```

In [8]:
```
# Check the descriptive statistics of numeric variables
telco_base_data.describe()
```

Out[8]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

SeniorCitizen is actually a categorical hence the 25%-50%-75% distribution is not propoer

75% customers have tenure less than 55 months

Average Monthly charges are USD 64.76 whereas 25% customers pay more than USD 89.85 per month

In [10]:
```
100*telco_base_data['Churn'].value_counts()/len(telco_base_data['Churn'])
```

Out[10]:
```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

In [11]:
```
telco_base_data['Churn'].value_counts()
```

Out[11]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

- Data is highly imbalanced, ratio = 73:27
- So we analyse the data with other features while taking the target values separately to get some insights.

In [12]:

```python
# Concise Summary of the dataframe, as we have too many columns, we are using the verbose = True mode
telco_base_data.info(verbose = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
customerID          7043 non-null object
gender              7043 non-null object
SeniorCitizen       7043 non-null int64
Partner             7043 non-null object
Dependents          7043 non-null object
tenure              7043 non-null int64
PhoneService        7043 non-null object
MultipleLines       7043 non-null object
InternetService     7043 non-null object
OnlineSecurity      7043 non-null object
OnlineBackup        7043 non-null object
DeviceProtection    7043 non-null object
TechSupport         7043 non-null object
StreamingTV         7043 non-null object
StreamingMovies     7043 non-null object
Contract            7043 non-null object
PaperlessBilling    7043 non-null object
PaymentMethod       7043 non-null object
MonthlyCharges      7043 non-null float64
TotalCharges        7043 non-null object
Churn               7043 non-null object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

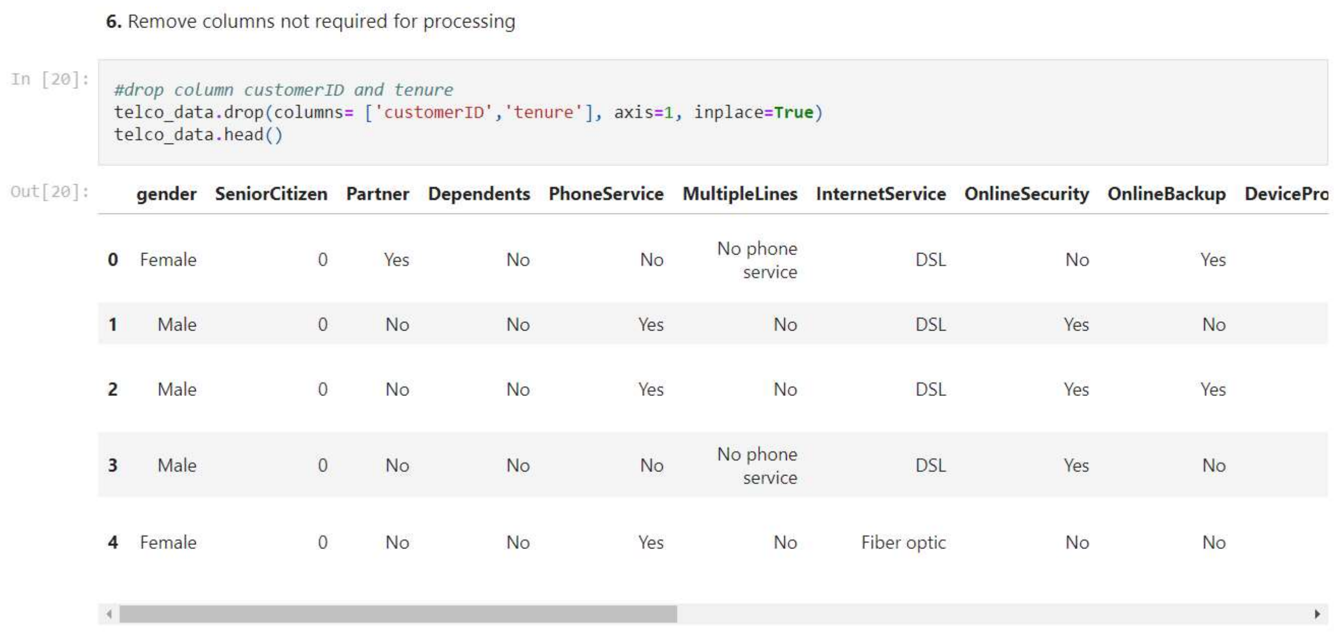
Data Cleaning

Data cleaning is a vital step in preparing data for machine learning. It involves identifying and rectifying various data quality issues, such as missing values, duplicates, outliers, and inconsistencies. This process is crucial for ensuring the accuracy and reliability of the dataset.

Handling missing data is a common challenge, and it requires decisions on imputation or removal. Duplicate records can distort model training and need to be detected and eliminated. Outliers, extreme data points, may negatively impact models and should be either removed or transformed.

Data transformation, including feature scaling and encoding, standardizes the data for machine learning algorithms. Consistency in data formatting, such as date or categorical variable standardization, is essential for coherence.

Data cleaning often requires domain knowledge to make informed decisions, and it should be well-documented for transparency and reproducibility in the machine learning process.



6. Remove columns not required for processing

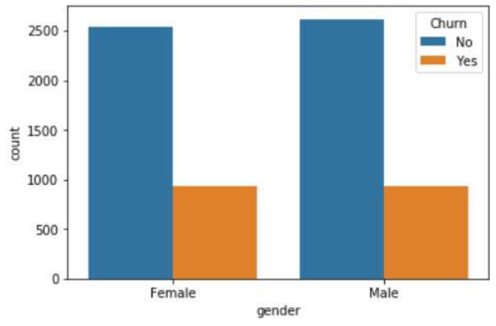
```
In [20]:
#drop column customerID and tenure
telco_data.drop(columns= ['customerID','tenure'], axis=1, inplace=True)
telco_data.head()
```

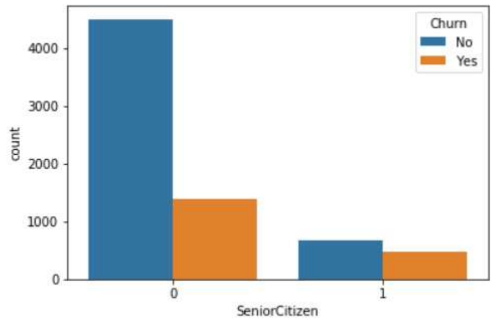
Out[20]:

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

Univariate analysis with visualization

```
In [20]:  for i, predictor in enumerate(telco_data.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
              plt.figure(i)
              sns.countplot(data=telco_data, x=predictor, hue='Churn')
```





Similarly for all columns, analysis is done- churn and not churned.

2. Convert the target variable 'Churn' in a binary numeric variable i.e. Yes=1 ; No = 0

```
In [21]:  telco_data['Churn'] = np.where(telco_data.Churn == 'Yes',1,0)

In [22]:  telco_data.head()
```

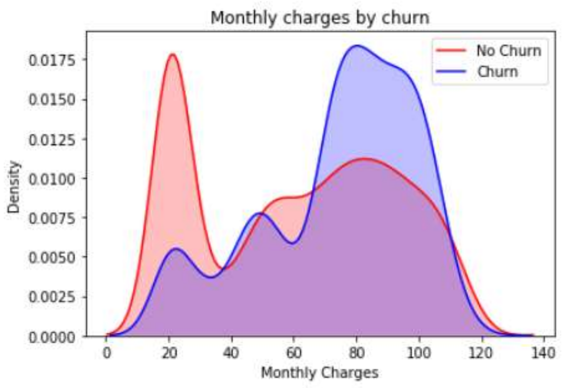
Out[22]:

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

10. Churn by Monthly Charges and Total Charges

In [25]:
```
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')
```

Out[25]:
```
Text(0.5, 1.0, 'Monthly charges by churn')
```



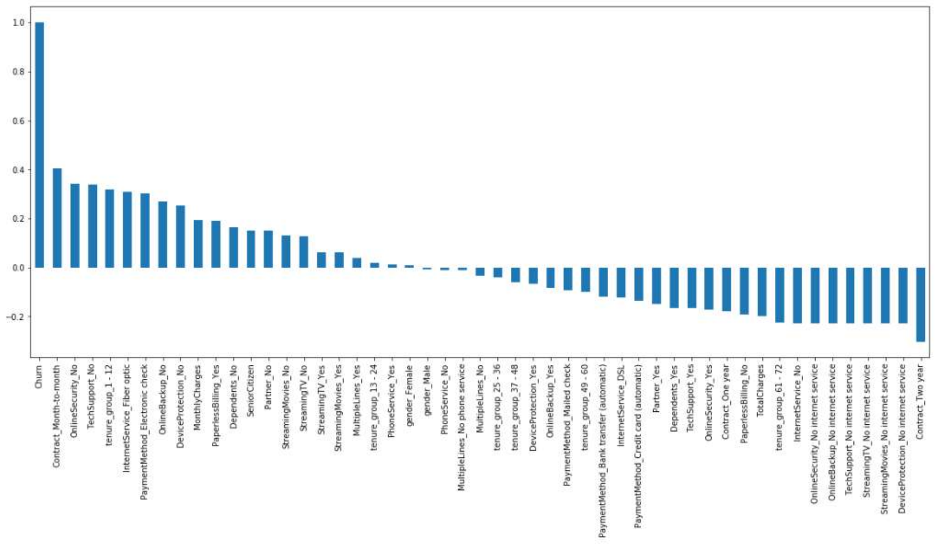
Insight: Churn is high when Monthly Charges ar high

In [26]:
```
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')
```

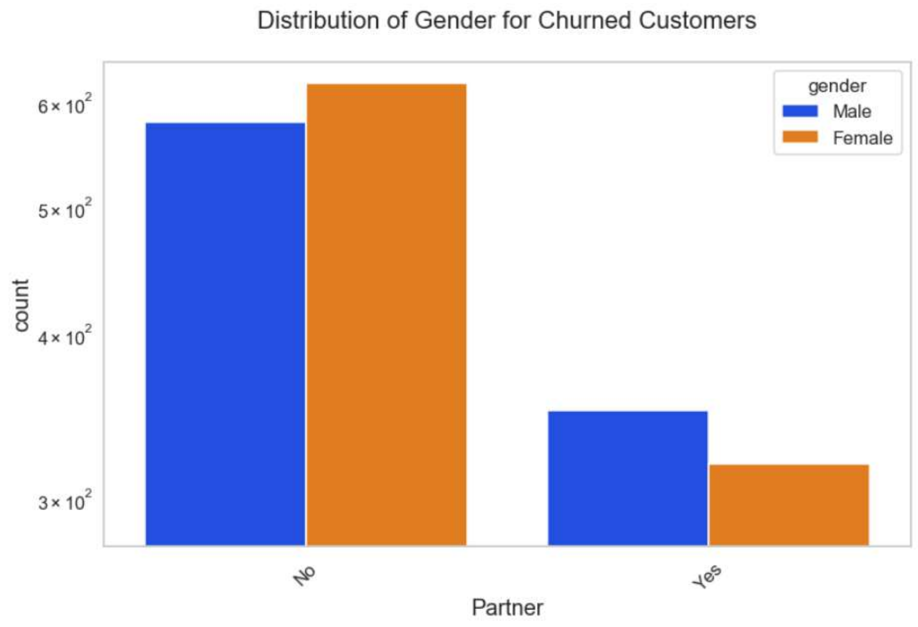
11. Build a corelation of all predictors with 'Churn'

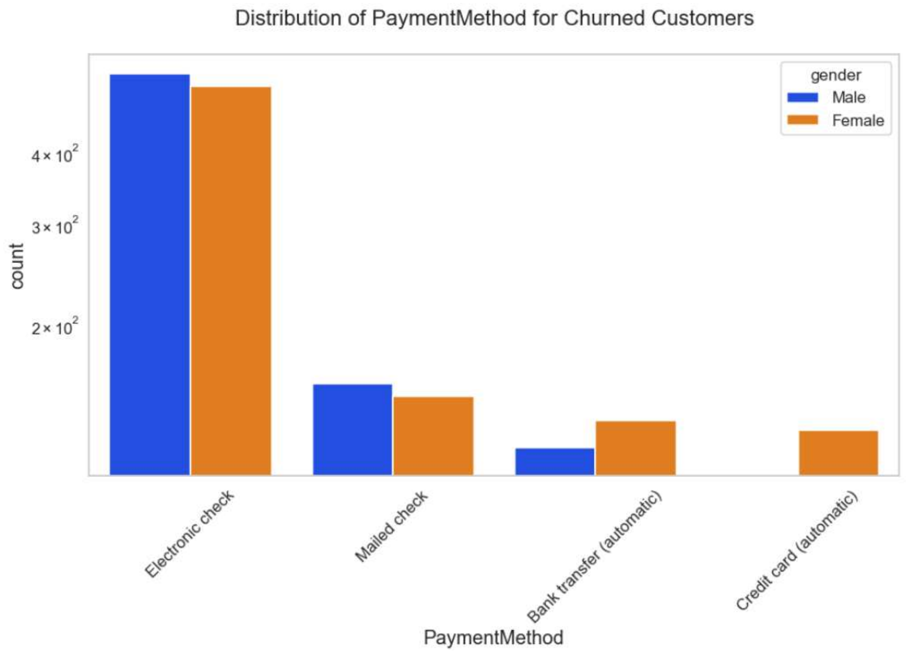
In [27]:
```
plt.figure(figsize=(20,8))
telco_data_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20d8a979f98>
```



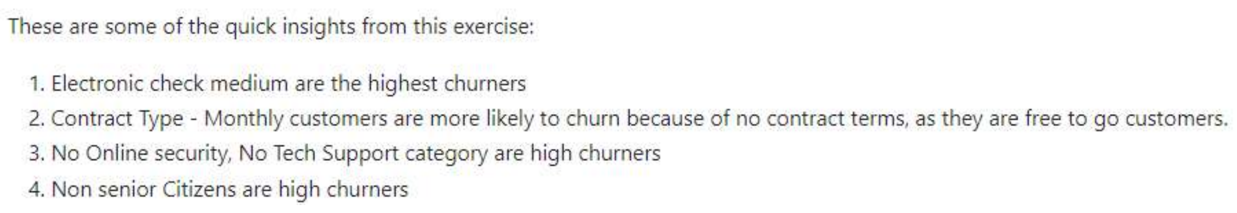
Bivariate analysis with visualization





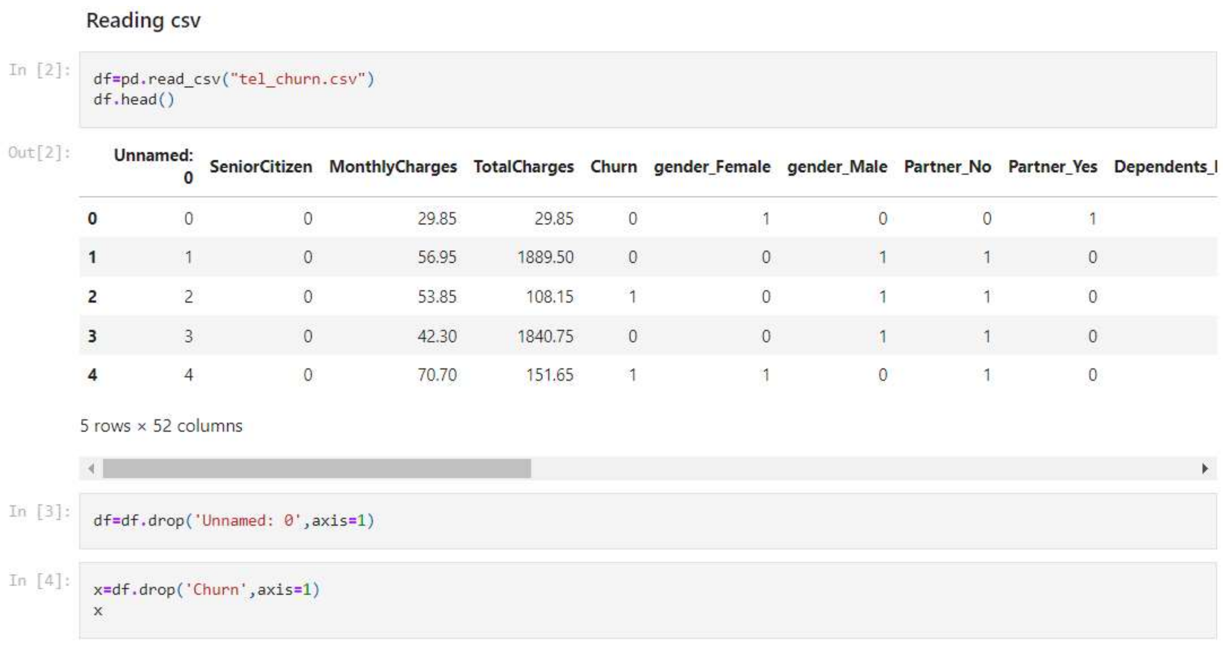
Similarly for every feature, this analysis is carried out and graph is plotted.

Conclusion about EDA


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

Model Building


Reading csv

```
In [2]: df=pd.read_csv("tel_churn.csv")
        df.head()
```

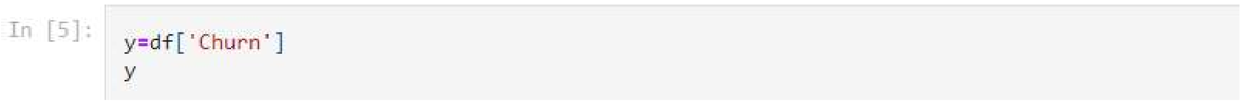
Out[2]:

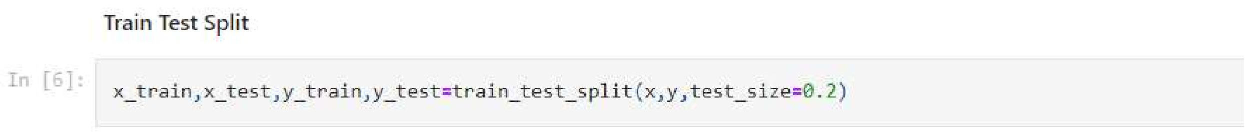
	Unnamed: 0	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_
0	0	0	29.85	29.85	0	1	0	0	1	
1	1	0	56.95	1889.50	0	0	1	1	0	
2	2	0	53.85	108.15	1	0	1	1	0	
3	3	0	42.30	1840.75	0	0	1	1	0	
4	4	0	70.70	151.65	1	1	0	1	0	

5 rows × 52 columns

```
In [3]: df=df.drop('Unnamed: 0',axis=1)
```

```
In [4]: x=df.drop('Churn',axis=1)
        x
```


```
In [5]: y=df['Churn']
        y
```


Train Test Split

```
In [6]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Decision TreeClassifier

A decision tree is a versatile machine learning algorithm used for classification and regression tasks. It represents a decision-making process through a tree-like structure. At the root node, the entire dataset is considered. The algorithm selects features that best split the data based on information gain or impurity reduction. Data is then divided into subsets, creating branches in the tree. This process recursively continues, with feature selection and splitting until stopping criteria, such as a maximum depth or minimum samples per leaf, are met.

Leaf nodes, the terminal points of the tree, represent the final decision. In classification, each leaf node signifies a class label, while in regression, it represents a predicted value. To make predictions for new data, you follow the tree from the root node down the branches, making decisions based on feature values until a leaf node is reached, which provides the final prediction.

Decision trees are valued for their interpretability and ability to handle various data types, including both categorical and numerical features. Nevertheless, they can overfit by creating overly complex trees that fit training data too closely. Techniques like pruning and ensemble methods like Random Forests can mitigate overfitting and enhance decision tree performance. Overall, decision trees are a fundamental tool in machine learning, finding applications in diverse fields such as finance, healthcare, and recommendation systems.

```
In [7]:  model_dt=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

```
In [8]:  model_dt.fit(x_train,y_train)

Out[8]:  DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
In [9]:  y_pred=model_dt.predict(x_test)
         y_pred

Out[9]:  array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

```
In [10]:  model_dt.score(x_test,y_test)

Out[10]:  0.7818052594171997
```

```
In [11]:  print(classification_report(y_test, y_pred, labels=[0,1]))

              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)

```
In [12]:  sm = SMOTEENN()
          X_resampled, y_resampled = sm.fit_sample(x,y)
```

```
In [13]:  xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_size=0.2)
```

```
In [14]:  model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

```
In [15]:  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))

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
```

```
In [16]:  print(metrics.confusion_matrix(yr_test, yr_predict))

[[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.

Random forest Classifier

Random Forest is a powerful ensemble learning algorithm in machine learning that leverages multiple decision trees to make more accurate and robust predictions. It combines the strengths of individual decision trees while mitigating their weaknesses, such as overfitting.

In a Random Forest, a predefined number of decision trees are created. Each tree is trained on a bootstrapped subset of the training data (a random sample with replacement). Additionally, during the creation of each tree, a random subset of features is considered for splitting at each node. This randomness injects diversity into the forest, making the individual trees different from one another.

When it's time to make a prediction, each tree in the Random Forest provides its own prediction (classification or regression). In classification tasks, the final prediction is often determined by a majority vote among the individual trees, while in regression tasks, it's the average of their predictions.

Random Forests offer several advantages, including high accuracy, robustness against overfitting, and the ability to handle large and complex datasets. They are also capable of feature importance ranking, which helps identify the most influential features in the data.

```
In [17]: from sklearn.ensemble import RandomForestClassifier
```

```
In [18]: model_rf=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

```
In [19]: model_rf.fit(x_train,y_train)

Out[19]: RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
In [20]: y_pred=model_rf.predict(x_test)
```

```
In [21]: model_rf.score(x_test,y_test)

Out[21]: 0.7953091684434968
```

```
In [22]: print(classification_report(y_test, y_pred, labels=[0,1]))

              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
```

```
In [23]: sm = SMOTEENN()
         X_resampled1, y_resampled1 = sm.fit_sample(x,y)
```

```
In [24]: xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(X_resampled1, y_resampled1,test_size=0.2)
```

```
In [25]: model_rf_smote=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=
```

```
In [26]: model_rf_smote.fit(xr_train1,yr_train1)

Out[26]: RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
In [27]: yr_predict1 = model_rf_smote.predict(xr_test1)
```

```
In [28]: model_score_r1 = model_rf_smote.score(xr_test1, yr_test1)
```

```
In [29]: print(model_score_r1)
         print(metrics.classification_report(yr_test1, yr_predict1))

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
```

```
In [30]: print(metrics.confusion_matrix(yr_test1, yr_predict1))
```

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

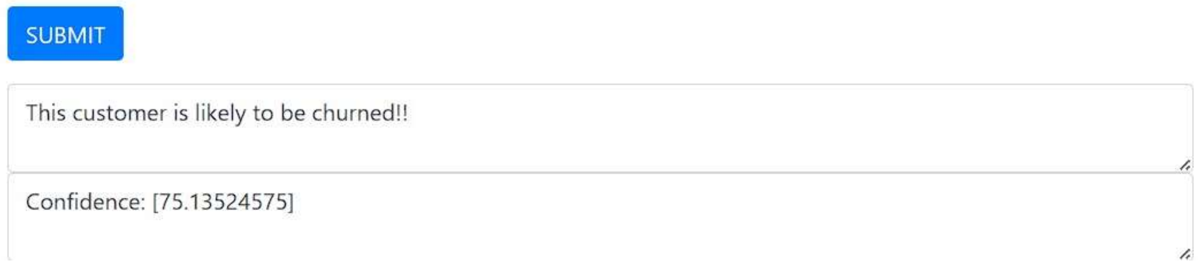
With RF Classifier, also we are able to get quite good results, infact better than Decision Tree.

Pickling the model

```
In [37]: import pickle

In [38]: filename = 'model.sav'

In [39]: pickle.dump(model_rf_smote, open(filename, 'wb'))

In [40]: load_model = pickle.load(open(filename, 'rb'))

In [41]: model_score_r1 = load_model.score(xr_test1, yr_test1)

In [42]: model_score_r1

Out[42]: 0.9427350427350427
```

Discussion and Conclusion

Output:



```
This customer is likely to be churned!!

Confidence: [75.13524575]
```

Conclusion:

In conclusion, the results of our Customer Churn Detection project are indeed promising, with the highest achieved accuracy rate of 94.27% by using Random Forest Classifier. This remarkable accuracy demonstrates the effectiveness of our detection model in distinguishing between churn and unchurned customers.

Certainly, here's an example of a conclusion for a customer churn prediction project. The customer churn prediction project has yielded valuable insights and actionable strategies to address the challenge of customer attrition. This project was motivated by the imperative need to reduce churn's impact on profitability, enhance resource efficiency, and maintain customer satisfaction.

Through the development of a predictive model, we have made significant strides in our approach to customer retention. The model accurately identifies customers at risk of churning, providing a data-driven framework for proactively targeting at-risk customers with tailored retention strategies.

The impact of this project extends beyond financial gains; it is about fostering customer loyalty, optimizing resource allocation, and strengthening our competitive position in the market. As we implement the model and recommendations into our operations, we look forward to witnessing the positive outcomes in terms of reduced churn rates, improved customer satisfaction, and an enhanced overall business performance. The journey towards effective customer churn management is an ongoing one, and with continued dedication to data-driven decision-making, we are well-positioned to stay ahead in the ever-evolving business landscape.

References:

[1] Journal of Bigdata
https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0191-6

[2] Research Gate
https://www.researchgate.net/publication/353098175_CUSTOMER_CHURN_PREDICTION

[3] Scientific Research Publishing
https://www.scirp.org/journal/paperinformation.aspx?paperid=96177

[4] IEEE Explore
https://ieeexplore.ieee.org/document/8538420

[5] Science Direct
https://www.sciencedirect.com/science/article/pii/S2666603023000143