```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as mtic
import matplotlib.pyplot as plot
```

```
telecom="C:\\Users\\Hp\\Downloads\\churn_data.csv"
```

```
telecom = pd.read_csv("C:\\Users\\Hp\\Downloads\\churn_data.csv", na_values = " ")
telecom.head()
```

	customerID	tenure	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Chur
0	7590-VHVEG	1	No	Month-to-month	Yes	Electronic check	29.85	29.85	N
1	5575-GNVDE	34	Yes	One year	No	Mailed check	56.95	1889.50	N
2	3668-QPYBK	2	Yes	Month-to-month	Yes	Mailed check	53.85	108.15	Ye
	7795-					Bank transfer			

```
customer_data = pd.read_csv("C:\\Users\\Hp\\Downloads\\churn_data.csv")
customer_data.head()
```

	customerID	tenure	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Chur
0	7590-VHVEG	1	No	Month-to-month	Yes	Electronic check	29.85	29.85	N
1	5575-GNVDE	34	Yes	One year	No	Mailed check	56.95	1889.5	N
2	3668-QPYBK	2	Yes	Month-to-month	Yes	Mailed check	53.85	108.15	Ye
	7795-					Bank transfer			

```
internet_data = pd.read_csv("C:\\Users\\Hp\\Downloads\\churn_data.csv")
internet_data.head()
```

	customerID	tenure	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Chur
0	7590-VHVEG	1	No	Month-to-month	Yes	Electronic check	29.85	29.85	N
1	5575-GNVDE	34	Yes	One year	No	Mailed check	56.95	1889.5	N
2	3668-QPYBK	2	Yes	Month-to-month	Yes	Mailed check	53.85	108.15	Ye
	7795-					Bank transfer			

```
# Merging on 'customerID'
df_1 = pd.merge(telecom, customer_data, how='inner', on='customerID')
```

```
# Final dataframe with all predictor variables
telecom1 = pd.merge(df_1, internet_data, how='inner', on='customerID')
```

```
# Let's see the head of our master dataset
telecom.head()
```

	customerID	tenure	PhoneService	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Chur
0	7590-VHVEG	1	No	Month-to-month	Yes	Electronic check	29.85	29.85	N
1	5575-GNVDE	34	Yes	One year	No	Mailed check	56.95	1889.50	N
2	3668-QPYBK	2	Yes	Month-to-month	Yes	Mailed check	53.85	108.15	Ye
	7795-					Bank transfer			

```
telecom.columns.values
```

```
array(['customerID', 'tenure', 'PhoneService', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

```
# Checking the data types of all the columns
telecom.dtypes
```

```
customerID           object
tenure                int64
PhoneService         object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                object
dtype: object
```

```
# Now lets explore if is there any missing or null values
telecom.TotalCharges = pd.to_numeric(telecom.TotalCharges, errors='coerce')
telecom.isna().any() # All False confirm there is no missing values
```

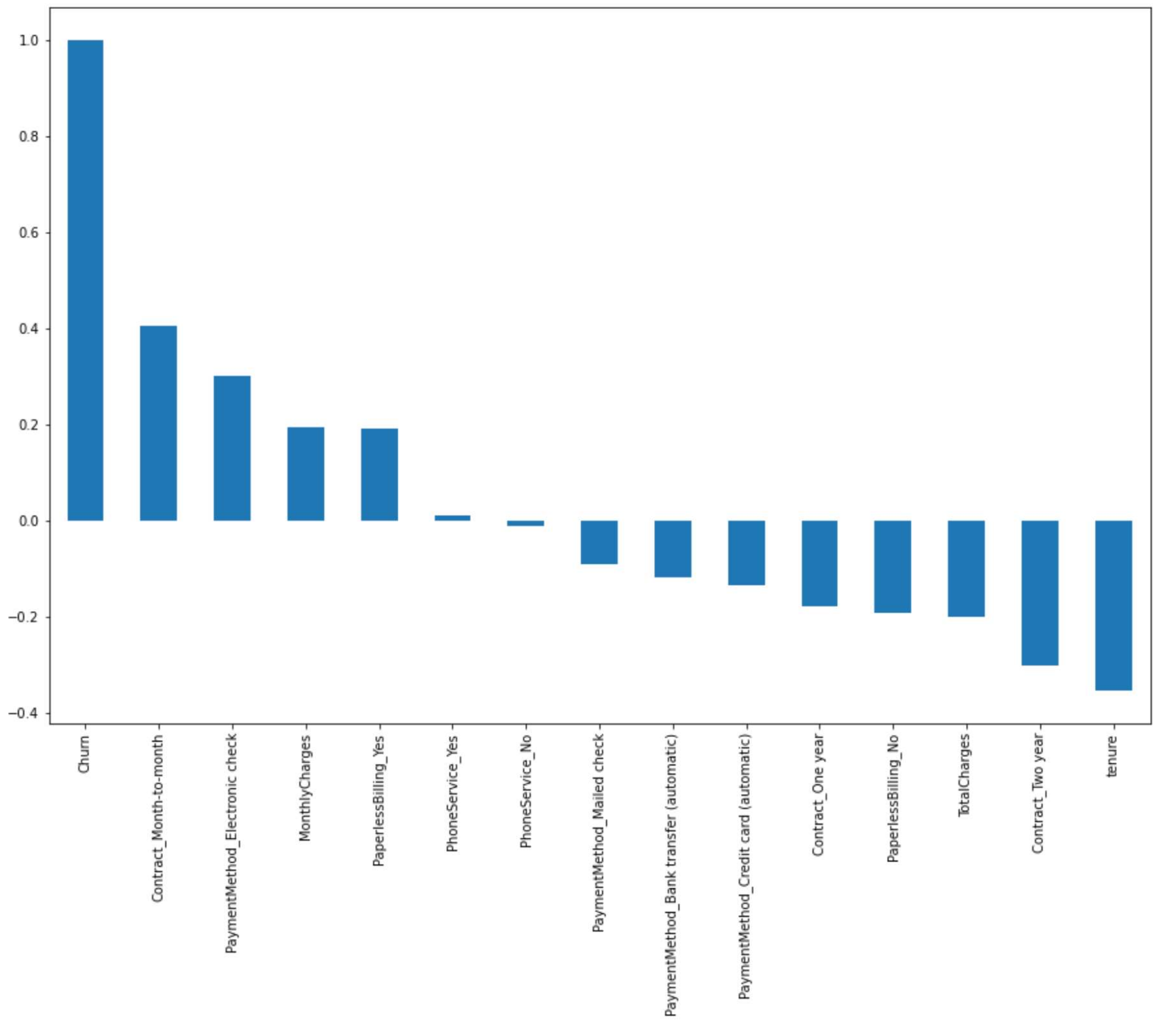
```
customerID          False
tenure              False
PhoneService        False
Contract            False
PaperlessBilling    False
PaymentMethod       False
MonthlyCharges      False
TotalCharges         True
Churn               False
dtype: bool
```

```
# Preprocessing
telecom.isnull().sum()
# There are 11 missing value for Total Charges, lets remove these 11 values having missing data from dataset
# Remove NA values
telecom.dropna(inplace = True)
# Lets remove customerId from dataset, which is not required for model
telecomdummy = telecom.iloc[:,1:]
# Converting Label variable i'e Churn to binary Numerical
telecomdummy['Churn'].replace(to_replace='No',value=0,inplace=True)
telecomdummy['Churn'].replace(to_replace='Yes',value=1,inplace=True)

# Convert categorical variable into dummy/indicator variables
# pd.get_dummies creates a new dataframe which consists of zeros and ones.
dummiesDf = pd.get_dummies(telecomdummy)
dummiesDf.head(20)
```

	tenure	MonthlyCharges	TotalCharges	Churn	PhoneService_No	PhoneService_Yes	Contract_Month-to-month	Contract_One year
0	1	29.85	29.85	0	1	0	1	0
1	34	56.95	1889.50	0	0	1	0	1
2	2	53.85	108.15	1	0	1	1	0
3	45	42.30	1840.75	0	1	0	0	1
4	2	70.70	151.65	1	0	1	1	0

```
# Feature Selection

# Now Lets check correlation of Churn with other variables
import matplotlib.pyplot as plt
plt.figure(figsize=(15,10))
dummiesDf.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

```
<AxesSubplot:>
```




```
#  Conclusion:  As per correlation, Month to month contracts, absence of online security and tech support seem to be positively correlated wi
#  While, tenure, two year contracts and Internet Service seem to be negatively correlated with churn.
# services such as Online security, streaming TV, online backup, tech support, Device protection, Partner and Streaming movies without intern

Y = dummiesDf['Churn'].values
#Accuracy 79.95
X = dummiesDf.drop(columns = ['Churn'])
# Accuracy 78.31%
#selected_features = ['Contract_Month-to-month','tenure','TotalCharges']
#Accuracy 79.31%
selected_features =['tenure', 'MonthlyCharges',
       'TotalCharges']
#Accuracy 76.46%
```

```
#selected_features=['Contract_Month-to-month','OnlineSecurity_No','TechSupport_No','tenure','Contract_Two year']
#Accuracy 79.53%
#selected_features=X.drop(columns=['PhoneService_Yes','gender_Female','gender_Male','PhoneService_No']).columns.values
X_select = X[selected_features]
# Lets scale all the variables from a range of 0 to 1
# Transforms features by scaling each feature to a given range.
#This estimator scales and translates each feature individually such that it is in the given range on the training set (0,1).
from sklearn.preprocessing import MinMaxScaler
features = X.columns.values
scaler = MinMaxScaler(feature_range=(0,1))
scaler.fit(X)
X = pd.DataFrame(scaler.transform(X))
X.columns = features

# Selected features
scaler.fit(X_select)
X_select = pd.DataFrame(scaler.transform(X_select))
X_select.columns=selected_features
X_select.head(20)
```

	tenure	MonthlyCharges	TotalCharges
0	0.000000	0.115423	0.001275
1	0.464789	0.385075	0.215867
2	0.014085	0.354229	0.010310
3	0.619718	0.239303	0.210241
4	0.014085	0.521891	0.015330
5	0.098592	0.809950	0.092511
6	0.295775	0.704975	0.222779
7	0.126761	0.114428	0.032668
8	0.380282	0.861194	0.349325
9	0.859155	0.377114	0.400317
10	0.169014	0.315423	0.065619
11	0.211268	0.006965	0.035541
12	0.802817	0.816915	0.653393
13	0.676056	0.850249	0.578987
14	0.338028	0.868159	0.307783
15	0.957746	0.945274	0.908880
16	0.718310	0.023881	0.115872
17	0.985915	0.880100	0.849694
18	0.126761	0.367662	0.058799
19	0.281690	0.714428	0.212797

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics


X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.2, random_state=99)
from sklearn.svm import SVC

modelSVM = SVC(kernel='linear')
modelSVM.fit(X_train,y_train)
preds = modelSVM.predict(X_test)
metrics.accuracy_score(y_test, preds)
```

```
0.8130774697938877
```

```
# Create the Confusion matrix for SVM
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,preds))
```

```
[[946  96]
 [167 198]]
```