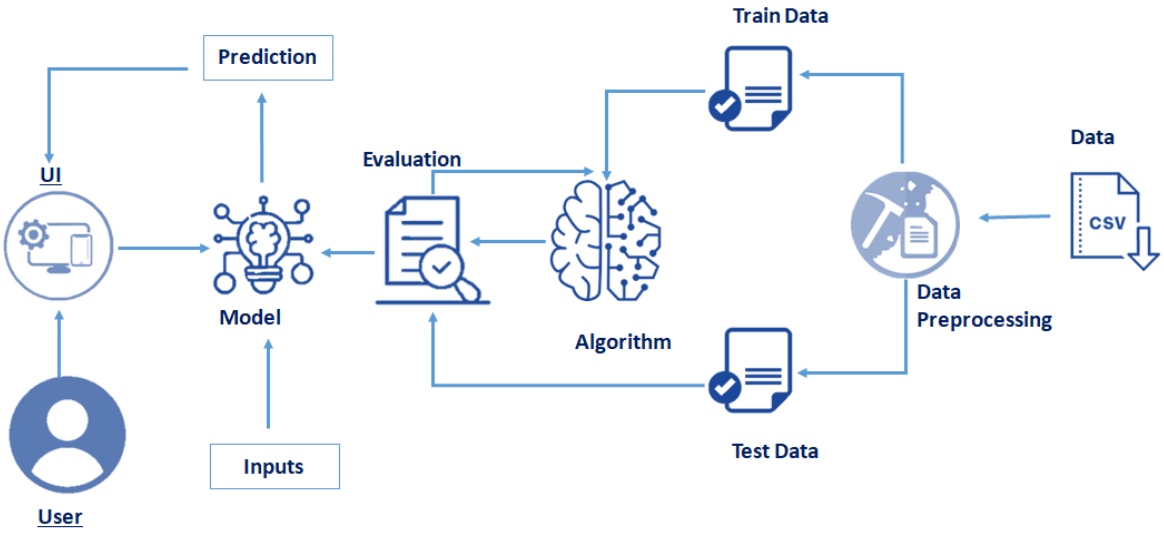
Telecom Customer Churn Prediction

Project Description:

Customer churn has become highly important for companies because of increasing competition among companies, increased importance of marketing strategies and conscious behaviour of customers in the recent years. Customers can easily trend toward alternative services. Companies must develop various strategies to prevent these possible trends, depending on the services they provide. During the estimation of possible churns, data from the previous churns might be used. An efficient churn predictive model benefits companies in many ways. Early identification of customers likely to leave may help to build cost effective ways in marketing strategies. Customer retention campaigns might be limited to selected customers but it should cover most of the customer. Incorrect predictions could result in a company losing profits because of the discounts offered to continuous subscribers.

Telecommunication industry always suffers from a very high churn rates when one industry offers a better plan than the previous there is a high possibility of the customer churning from the present due to a better plan in such a scenario it is very difficult to avoid losses but through prediction we can keep it to a minimal level. A machine learning model is builtand this helps to identity the probable churn customers and then makes the necessary business decisions.

Technical Architecture:



Pre requisites:

To complete this project, you must require following software's concepts and packages

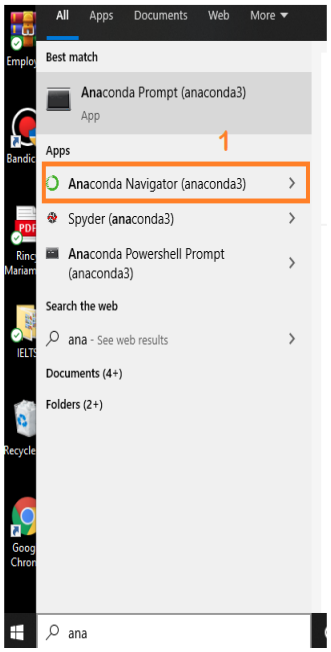
- **Anaconda navigator:**
  - Refer to the link below to download anaconda navigator
  - **Link : https://www.youtube.com/watch?v=5mDYijMfSzs**
- **Python packages:**

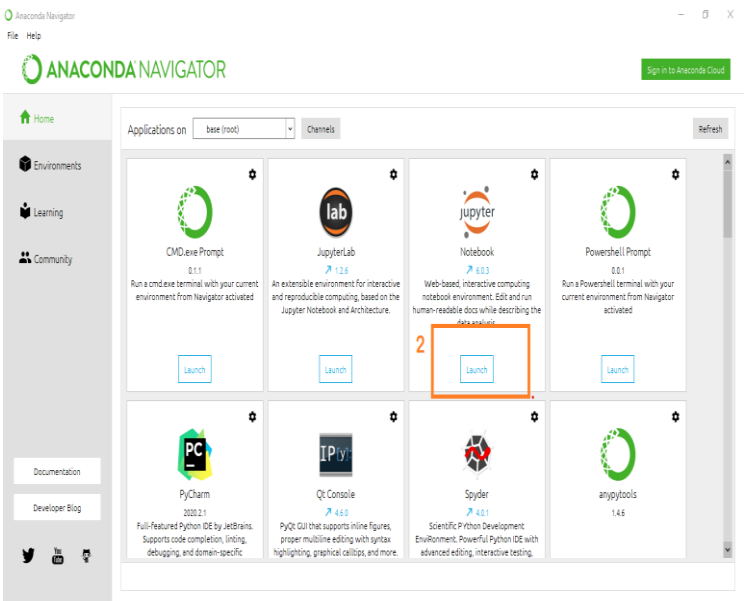
Open anaconda prompt as administrator.

- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install matplotlib” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install Flask” and click enter.

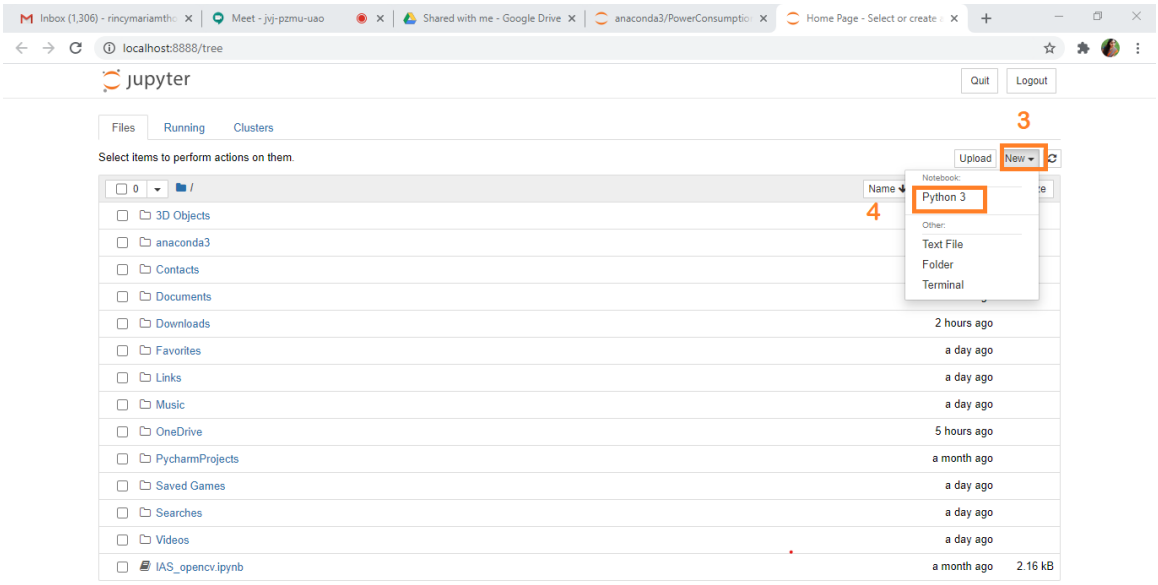
The above steps allow you to install the packages in the anaconda environment

- **Launch Jupyter**
  - Search for Anaconda Navigator and open Launch Jupyter notebook.





- Then you will be able to see that the jupyter notebook runs on local host:8888.
- To Create a new file Go to New →Python3.The file in jupyter notebook is saved with .ipynb extension.



- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process / clean the data using different data preprocessing techniques.
- You will able to analyse or get insights of data through visualization.
- Applying different algorithms according to dataset and based on visualization.
- You will able to know how to find accuracy of the model.
- You will be able to know how to build a web application using Flask framework.

Project Flow:

- User interacts with the UI (User Interface) to enter the input values
- Entered input values are analyzed by the model which is integrated
- Once model analyses the input the prediction is showcased on the UI

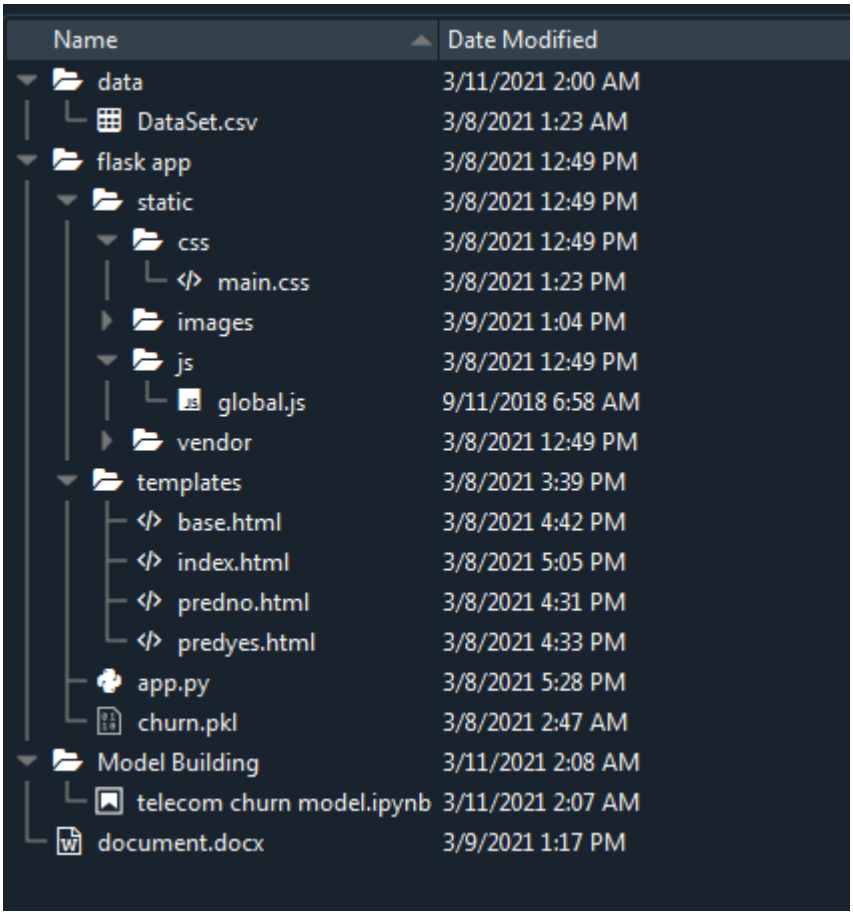
To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
  - Collect the dataset or Create the dataset

- Data Preprocessing.
    - Import the Libraries.
    - Importing the dataset.
    - Checking for Null Values.
    - Data Visualization.
    - Taking care of Missing Data.
    - Label Encoding.
    - OneHot Encoding.
    - Splitting Data into Train and Test.
    - Feature Scaling.
- Model Building
    - Import the model building Libraries
    - Initializing the model
    - Training and testing the model
    - Evaluation of Model
    - Save the Model
- Application Building
    - Create an HTML file
    - Build a Python Code

Project Structure:

Create a Project folder which contains files as shown below



- A python file called app.py for server side scipting.

- We need the model which is saved and the saved model in this content is **churn.pkl**
- Templates folder which contains base.HTML file, index.HTML file, predyes.HTML , predno.HTML file.
- Static folder which contains css folder which contains main.css , js folder which contains global.js , images folder and vendor folder.

Milestone 1: Data Collection:

ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training **data set.** It is the actual **data set** used to train the model for performing various actions.

Activity1: Download The dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

The dataset used for this project was obtained from Kaggle . Please refer to the link given below to download the data set and to know about the dataset

https://www.kaggle.com/shrutimechlearn/churn-modelling

Milestone 2: Data Preprocessing

Data Pre-processing includes the following main tasks

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Label Encoding.
- OneHot Encoding.
- Splitting Data into Train and Test.
- Feature Scaling.

Activity 1: Import Necessary Libraries

- It is important to import all the necessary libraries such as pandas, numpy, matplotlib.
- **Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
- **Pandas**- It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

- **Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Matplotlib**- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python

```
#import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Activity 2: Importing the Dataset

- You might have your data in .csv files, .excel files

- Let's load a .csv data file into pandas using **read_csv() function.**We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).

```
#import dataset
data = pd.read_csv(r"C:\Users\ACER\Desktop\data science\Telecom churn modelling\data\DataSet.csv")
```

- If your dataset is in some other location ,Then

  Data=pd.read_csv(r"File_location")

  Note:r stands for "raw" and will cause backslashes in the string to be interpreted as actual backslashes rather than special characters.

- If the dataset in same directory of your program, you can directly read it, without giving raw as r.

- Our Dataset churndata contains following Columns

  1. gender
  2. SeniorCitizen
  3. Partner
  4. Dependents
  5. tenure
  6. PhoneService
  7. MultipleLines
  8. InternetService
  9. OnlineSecurity
  10. OnlineBackup
  11. DeviceProtection

12. TechSupport
13. StreamingTV
14. StreamingMovies
15. Contract
16. PaperlessBilling
17. PaymentMethod
18. MonthlyCharges
19. TotalCharges
20. Churn

The output column to be predicted is **Churn** .Based on the input variables we predict the financial risk.

Activity 3: Analyse the data

- head() method is used to return top n (5 by default) rows of a DataFrame or series.

```
data.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBa
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
1	Male	0	No	No	34	Yes	No	DSL	Yes	No
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No

- describe() method computes a summary of statistics like count, mean, standard deviation, min, max and quartile values.

```
data.describe()
```

The output is as shown below

	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

- info() gives information about the data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7043 non-null   object 
 1   SeniorCitizen     7043 non-null   int64  
 2   Partner           7043 non-null   object 
 3   Dependents        7043 non-null   object 
 4   tenure            7043 non-null   int64  
 5   PhoneService      7043 non-null   object 
 6   MultipleLines     7043 non-null   object 
 7   InternetService   7043 non-null   object 
 8   OnlineSecurity    7043 non-null   object 
 9   OnlineBackup      7043 non-null   object 
 10  DeviceProtection  7043 non-null   object 
 11  TechSupport       7043 non-null   object 
 12  StreamingTV       7043 non-null   object 
 13  StreamingMovies   7043 non-null   object 
 14  Contract          7043 non-null   object 
 15  PaperlessBilling  7043 non-null   object 
 16  PaymentMethod     7043 non-null   object 
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   object 
 19  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB
```

Activity 4: Handling Missing Values

1. After loading it is important to check the complete information of data as it can indication many of the hidden infomation such as null values in a column or a row

2.Check whether any null values are there or not. if it is present then following can be done,

a.Imputing data using Imputation method in sklearn

b.Filling NaN values with mean, median and mode using fillna() method.

```
#checking for null values
data.TotalCharges = pd.to_numeric(data.TotalCharges, errors='coerce')
data.isnull().any()
```

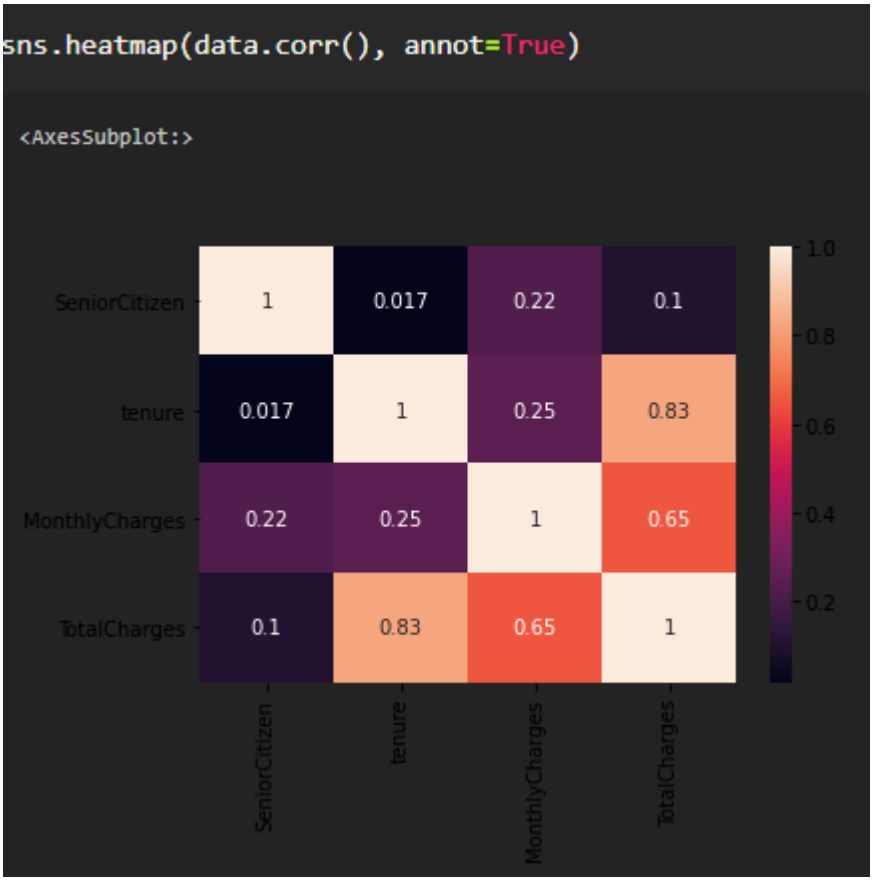
```
gender             False
SeniorCitizen      False
Partner            False
Dependents         False
tenure             False
PhoneService       False
MultipleLines      False
InternetService    False
OnlineSecurity     False
OnlineBackup       False
DeviceProtection   False
TechSupport        False
StreamingTV        False
StreamingMovies    False
Contract           False
PaperlessBilling   False
PaymentMethod      False
MonthlyCharges     False
TotalCharges        True
Churn              False
dtype: bool
```

```
data["TotalCharges"].fillna(data["TotalCharges"].median() , inplace =True)
```

```
data.isnull().sum()
```

```
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       0
Churn              0
dtype: int64
```

3. Heatmap:It is way of representing the data in 2-D form.It gives coloured visual summary of the data



From the heatmap, we see that there are no missing values in the dataset

Activity 5: Data Visualisation

- Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data.
- Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.
- To visualize the dataset we need libraries called Matplotlib and Seaborn.
- The Matplotlib library is a Python 2D plotting library which allows you to generate plots, scatter plots, histograms, bar charts etc.

Let's visualize our data using Matplotlib and searborn library.

Before diving into the code, let's look at some of the basic properties we will be using when plotting.

xlabel: Set the label for the x-axis.

ylabel: Set the label for the y-axis.

title: Set a title for the axes.

Legend: Place a legend on the axes.

1. data.corr() gives the correlation between the columns

```
data.corr()
```

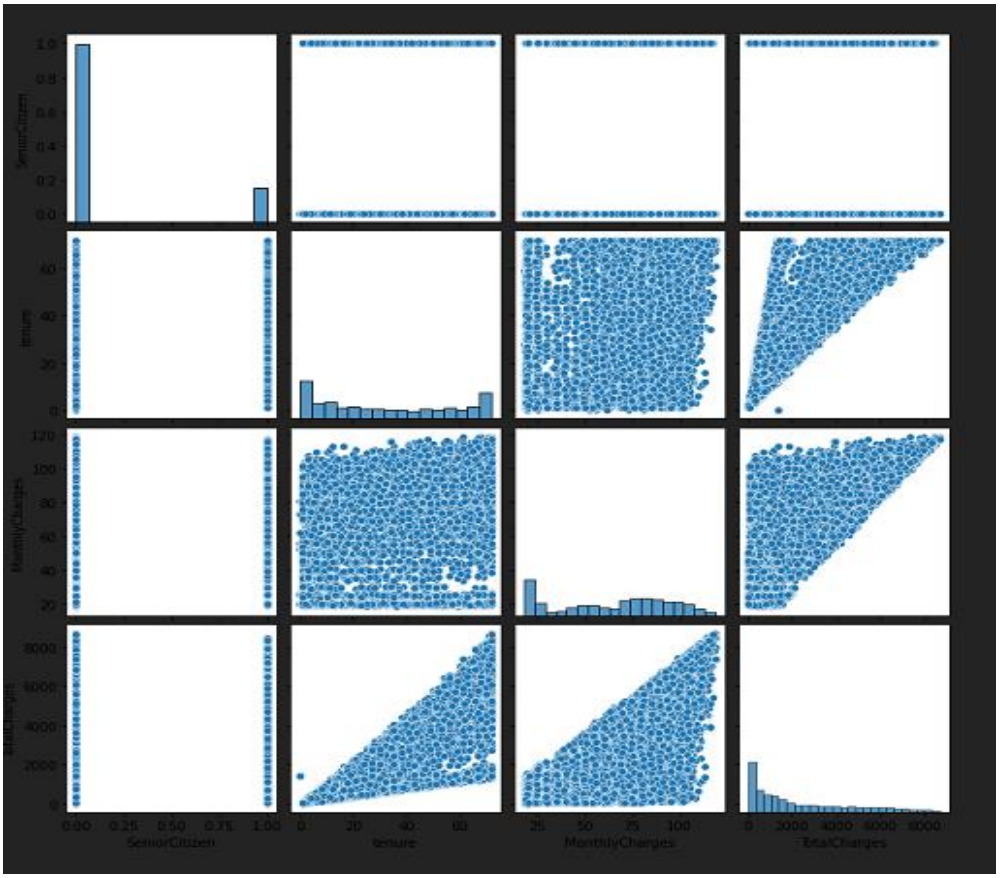
	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
SeniorCitizen	1.000000	0.016567	0.220173	0.102652
tenure	0.016567	1.000000	0.247900	0.825464
MonthlyCharges	0.220173	0.247900	1.000000	0.650864
TotalCharges	0.102652	0.825464	0.650864	1.000000

2. Pair Plot: Plot pairwise relationships in a dataset.

- By default, this function will create a grid of Axes such that each numeric variable in data will by shared across the y-axes across a single row and the x-axes across a single column. The diagonal plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.
- We implement this using the below code

```
sns.pairplot(data=data, markers=["^","v"], palette="inferno")
```

The output is as shown below



Pair plot usually gives pair wise relationships of the columns in the dataset

Activity 6: Label Encoding

In machine learning, we usually deal with datasets which contains multiple labels in one or more than one columns. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labelled in words.

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data["gender"] = le.fit_transform(data["gender"])
data["Partner"] = le.fit_transform(data["Partner"])
data["Dependents"] = le.fit_transform(data["Dependents"])
data["PhoneService"] = le.fit_transform(data["PhoneService"])
data["MultipleLines"] = le.fit_transform(data["MultipleLines"])
data["InternetService"] = le.fit_transform(data["InternetService"])
data["OnlineSecurity"] = le.fit_transform(data["OnlineSecurity"])
data["OnlineBackup"] = le.fit_transform(data["OnlineBackup"])
data["DeviceProtection"] = le.fit_transform(data["DeviceProtection"])
data["TechSupport"] = le.fit_transform(data["TechSupport"])
data["StreamingTV"] = le.fit_transform(data["StreamingTV"])
data["StreamingMovies"] = le.fit_transform(data["StreamingMovies"])
data["Contract"] = le.fit_transform(data["Contract"])
data["PaperlessBilling"] = le.fit_transform(data["PaperlessBilling"])
data["PaymentMethod"] = le.fit_transform(data["PaymentMethod"])
data["Churn"] = le.fit_transform(data["Churn"])
```

Data after label encoding

```
data.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBa
0	0	0	1	0	1	0	1	0	0	2
1	1	0	0	0	34	1	0	0	2	0
2	1	0	0	0	2	1	0	0	2	2
3	1	0	0	0	45	0	1	0	2	0
4	0	0	0	0	2	1	0	1	0	0

All the data is converted into numerical values.

Activity 7: Splitting the Dataset into Dependent and Independent variable

- In machine learning, the concept of dependent variable (y) and independent variables(x) is important to understand. Here, Dependent variable is nothing but output in dataset and independent variable is all inputs in the dataset.
- With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

To read the columns, we will use **iloc** of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

Let's split our dataset into independent and dependent variables.

1. The independent variable in the dataset would be considered as 'x' and gender,SeniorCitizen,Partner,Dependents,tenure,PhoneService,MultipleLines,InternetService,OnlineSecurity,OnlineBackup,DeviceProtection,TechSupport,StreamingTV,StreamingMovies,Contract,PaperlessBilling,PaymentMethod,MonthlyCharges,TotalCharges columns would be considered as independent variable.
2. The dependent variable in the dataset would be considered as 'y' and the 'Churn' column is considered as dependent variable.

Now we will split the data of independent variables,

```
x= data.iloc[:,0:19].values
y= data.iloc[:,19:20].values
```

From the above code **":"** indicates that you are considering all the rows in the dataset and **"0:18"** indicates that you are considering columns 0 to 8 such as sex, job and purpose as input values and assigning them to variable x. In the same way in second line **":"** indicates you are considering all the rows and **"18:19"** indicates that you are considering only last column as output value and assigning them to variable y.

After splitting we see the data as below

x

```
x


 array([[0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 2.0000e+00, 2.9850e+01,
         2.9850e+01],
        [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.0000e+00, 5.6950e+01,
         1.8895e+03],
        [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.0000e+00, 5.3850e+01,
         1.0815e+02],
        ...,
        [0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 2.0000e+00, 2.9600e+01,
         3.4645e+02],
        [1.0000e+00, 1.0000e+00, 1.0000e+00, ..., 3.0000e+00, 7.4400e+01,
         3.0660e+02],
        [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 0.0000e+00, 1.0565e+02,
         6.8445e+03]])
```

Y

```
y
```

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

Activity 8: OneHot Encoding

Sometimes in datasets, we encounter columns that contain numbers of no specific order of preference. The data in the column usually denotes a category or value of the category and also when the data in the column is label encoded. This confuses the machine learning model, to avoid this, the data in the column should be One Hot encoded.

One Hot Encoding –

It refers to splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column. Each column contains “0” or “1” corresponding to which column it has been placed.

```
from sklearn.preprocessing import OneHotEncoder
one = OneHotEncoder()
a= one.fit_transform(x[:,6:7]).toarray()
b= one.fit_transform(x[:,7:8]).toarray()
c= one.fit_transform(x[:,8:9]).toarray()
d= one.fit_transform(x[:,9:10]).toarray()
e= one.fit_transform(x[:,10:11]).toarray()
f= one.fit_transform(x[:,11:12]).toarray()
g= one.fit_transform(x[:,12:13]).toarray()
h= one.fit_transform(x[:,13:14]).toarray()
i= one.fit_transform(x[:,14:15]).toarray()
j= one.fit_transform(x[:,16:17]).toarray()
x=np.delete(x,[6,7,8,9,10,11,12,13,14,16],axis=1)
x=np.concatenate((a,b,c,d,e,f,g,h,i,j,x),axis=1)
```

Activity 9: Splitting the data into Train and Test

- When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset

which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

- But the question is, how do you split the data? You can't possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is,**'train_test_split.'** Using this we can easily split the dataset into the training and the testing datasets in various proportions.
- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.
- In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.We will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.
- There are a few other parameters that we need to understand before we use the class:
- **test_size** — this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset
- **train_size** — you have to specify this parameter only if you're not specifying the test_size. This is the same as test_size, but instead you tell the class what percent of the dataset you want to split as the training set.
- **random_state** — here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random_state class, which will become the number generator. If you don't pass anything, the Random_state instance used by np.random will be used instead.
- Now split our dataset into train set and test using train_test_split class from scikit learn library.

```
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)
```

Activity 10: Feature Scaling

There is huge disparity between the x values so let us use feature scaling.

Feature scaling is a method used to normalize the range of independent variables or features of data.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

Milestone 3: Model Buiding:

Model building includes the following main tasks

- Import the model building Libraries
- Initializing the model
- Training and testing the model
- Evaluation of Model
- Save the Model

Activity 1: Training and Testing the Model

- Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.
- There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms are Regression algorithms.

  1.Logistic Regression

  2.Decision Tree Classifier

  3.Random Forest Classifier

  4.SVM

We're going to use x_train and y_train obtained above in train_test_split section to train our decision tree regression model. We're using the fit method and passing the parameters as shown below.

We are using the algorithm from Scikit learn library to build the model as shown below,

```
from sklearn.svm import SVC
svm = SVC(kernel = "linear")
svm.fit(x_train,y_train)
```

```
C:\Users\ACER\.conda\envs\tf\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)


SVC(kernel='linear')
```

```
svm_pred = svm.predict(x_test)

svm_pred

 array([0, 0, 0, ..., 0, 0, 0])
```

Once the model is trained, it's ready to make predictions. We can use the **predict** method on the model and pass **x_test** as a parameter to get the output as **y_pred.**

> Notice that the prediction output is an array of real numbers corresponding to the input array.

Testing the model with own values

```
svm_pred_own = svm.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,

svm_pred_own

 array([0])
```

Activity 2: Model Evaluation

After training the model, the model should be tested by using the test data which is been separated while splitting the data for checking the functionality of the model.

Regression Evaluation Metrics:

These model evaluation techniques are used to find out the accuracy of models built in classification type of machine learning models. We have three types of evaluation methods.

- Accuracy_score
- Confusion matrix
- Roc- Auc Curve

1. Accuracy_score

It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

2. Confusion Matrix

It is a matrix representation of the results of any binary testing

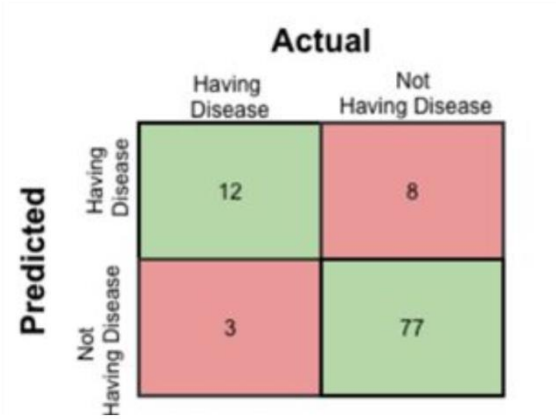


Fig: Confusion Matrix of prediction a disease

1. True Positive: 12 (You have predicted the positive case correctly!)
2. True Negative: 77 (You have predicted negative case correctly!)
3. False Positive: 8 (You have predicted these people as having disease, but in actual they do not have.)
4. False Negative: 3 (Wrong predictions)

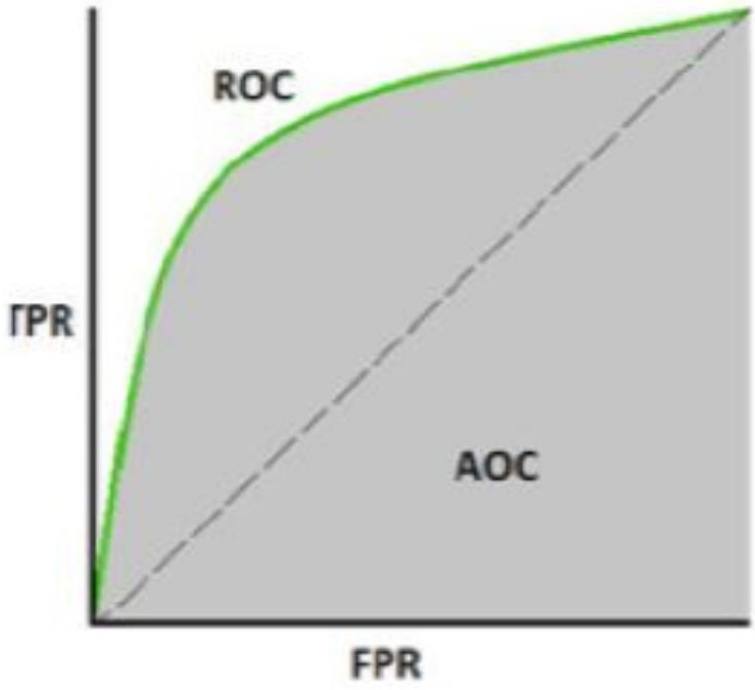
3. Roc-Auc Curve

AUC is the area under the ROC curve. AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes.

AUC - ROC curve is a performance measurement for classification problem at various thresholds settings

• ROC is a probability curve and AUC represents degree or measure of separability.

• Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1sThe ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.



For testing the model we use the below method,

```
svm_acc = accuracy_score(svm_pred,y_test)

svm_acc

 0.7977288857345636

svm_cm = confusion_matrix(svm_pred,y_test)

svm_cm

 array([[932, 176],
        [109, 192]], dtype=int64)
```

Activity 3: Save the Model

After building the model we have to save the model.

Pickle in **Python** is primarily **used** in serializing and deserializing a **Python** object structure. In other words, it's the process of converting a **Python** object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network. wb indicates write method and rd indicates read method.

This is done by the below code

```
import pickle
pickle.dump(svm,open("churn.pkl" , "wb"))
```

Milestone 4 : Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script

Activity 1: Build HTML Code

- In this HTML page, we will create the front end part of the web page. In this page we will accept input from the user and Predict the values.
  For more information regarding HTML
  https://www.w3schools.com/html/
  In our project we have 3 HTML files ,they are
  1. base.html
  2. index.html
  3. predyes.html
  4. predno.html

base.html

```
    <!-- Title Page-->
    <title>Telecom Churn Modelling</title>

    <!-- Icons font CSS-->
    <link href="{{ url_for('static', filename='vendor/font-awesome-4.7/css/font-awesome.min.css') }}" rel="stylesheet">
    <link href="{{ url_for('static', filename='vendor/mdi-font/css/material-design-iconic-font.min.css') }}" rel="stylesheet">

    <!-- Font special for pages-->
    <link href="https://fonts.googleapis.com/css?family=Roboto:100,100i,300,300i,400,400i,500,500i,700,700i,900,900i" rel="stylesheet">

    <!-- Vendor CSS-->
    <link href="{{ url_for('static', filename='vendor/select2/select2.min.css') }}" rel="stylesheet">

    <!-- Main CSS-->
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
</head>

<body>
    <div class="page-wrapper bg-red p-t-100 p-b-100 font-robo">
        <div class="wrapper wrapper--w960">
            <div class="card card-2">
                <div class="card-heading"></div>
                <div class="card-body">
                    <h2 class="title">Telecom Customer Churn Prediction</h2>
                    <div class="row row-space">

                        <p>Customer churn has become highly important for companies because of increasing
                            competition among companies, increased importance of marketing strategies and conscious
                            behaviour of customers in the recent years. Customers can easily trend toward alternative
                            services. Companies must develop various strategies to prevent these possible trends,
                            depending on the services they provide. During the estimation of possible churns, data from
                            the previous churns might be used. An efficient churn predictive model benefits companies in
                            many ways. Early identification of customers likely to leave may help to build cost effective
                            ways in marketing strategies. Customer retention campaigns might be limited to selected
                            customers but it should cover most of the customer. Incorrect predictions could result in a
                            company losing profits because of the discounts offered to continuous subscribers.</p>
                        <div class="p-t-30">
```

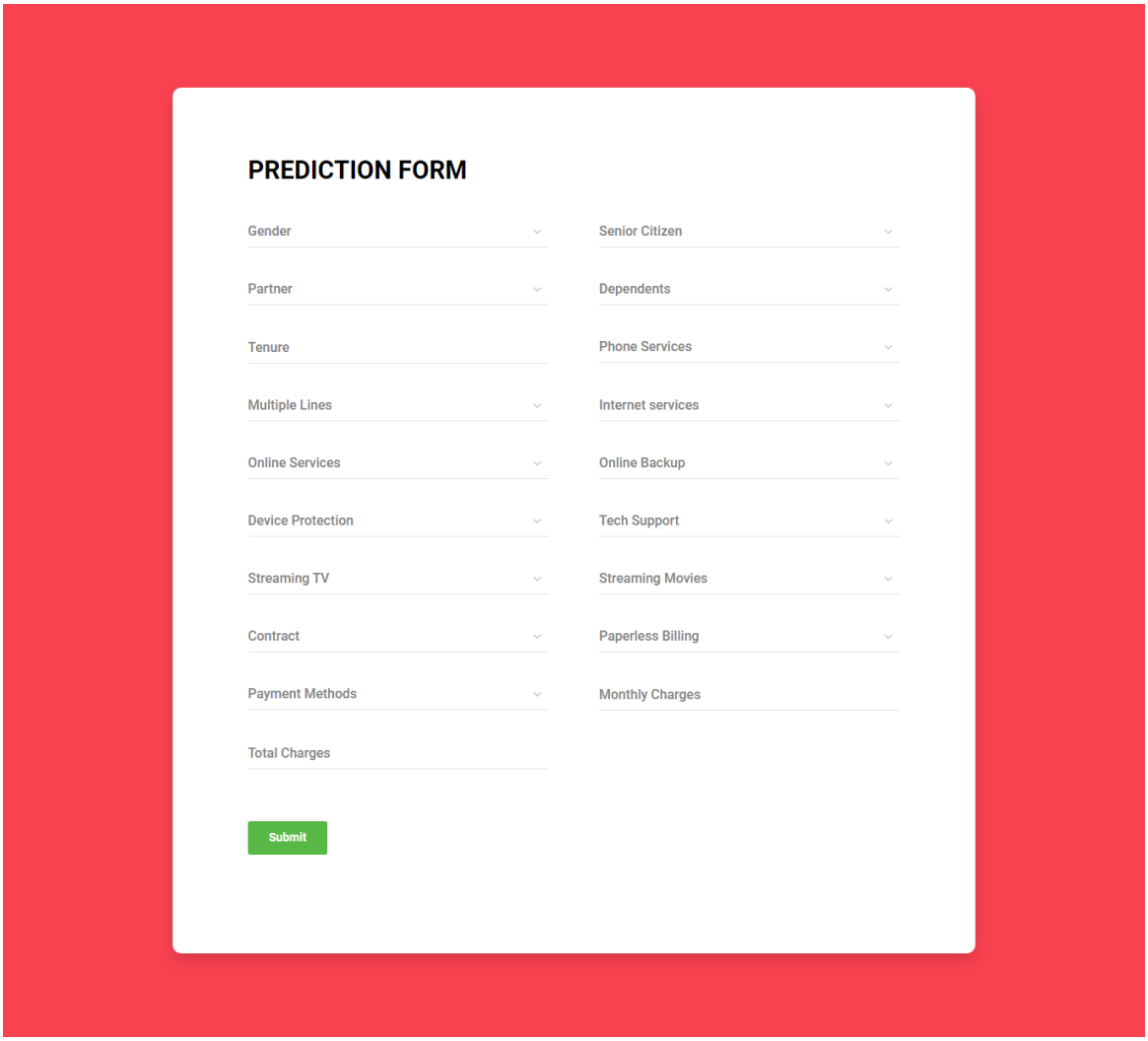
```
                        <div class="p-t-30">
                            <img src="../static/images/yes.jpg" width="450" height="200"  >
                        </div>
                        <div class="p-t-30">
                            <form action="/assesment">
                                <button class="btn btn--radius btn--green" type="submit">Click me to continue with prediction</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Jquery JS-->
    <script src="{{ url_for('static', filename='vendor/jquery/jquery.min.js') }}" type="text/javascript"></script>
    <!-- Vendor JS-->
    <script src="{{ url_for('static', filename='vendor/select2/select2.min.js') }}" type="text/javascript"></script>


    <!-- Main JS-->
    <script src="{{ url_for('static', filename='js/global.js') }}" type="text/javascript"></script>

</body>

</html>
<!-- end document-->
```

The html page looks like

index.html

```
    <!-- Title Page-->
    <title>Telecom Customer Churn Prediction</title>

    <!-- Icons font CSS-->
    <link href="{{ url_for('static', filename='vendor/font-awesome-4.7/css/font-awesome.min.css') }}" rel="stylesheet">
    <link href="{{ url_for('static', filename='vendor/mdi-font/css/material-design-iconic-font.min.css') }}" rel="stylesheet">

    <!-- Font special for pages-->
    <link href="https://fonts.googleapis.com/css?family=Roboto:100,100i,300,300i,400,400i,500,500i,700,700i,900,900i" rel="stylesheet">

    <!-- Vendor CSS-->
    <link href="{{ url_for('static', filename='vendor/select2/select2.min.css') }}" rel="stylesheet">

    <!-- Main CSS-->
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
</head>
<style>

</style>
<body>
    <div class="page-wrapper bg-red p-t-100 p-b-100 font-robo">
        <div class="wrapper wrapper--w960">
            <div class="card card-2">
                <div class="card-heading"></div>
                <div class="card-body">
                    <h2 class="title">Prediction form</h2>
                    <form action="/predict" method="post">
                        <div class="row row-space">
                            <div class="col-2">
                                <div class="input-group">
                                    <div class="rs-select2 js-select-simple select--no-search">
                                        <select name="gender">
                                            <option disabled="disabled" selected="selected">Gender</option>
                                            <option value="f">female</option>
                                            <option value="m">Male</option>
                                        </select>
                                        <div class="select-dropdown"></div>
```

The html page looks like



predyes.html

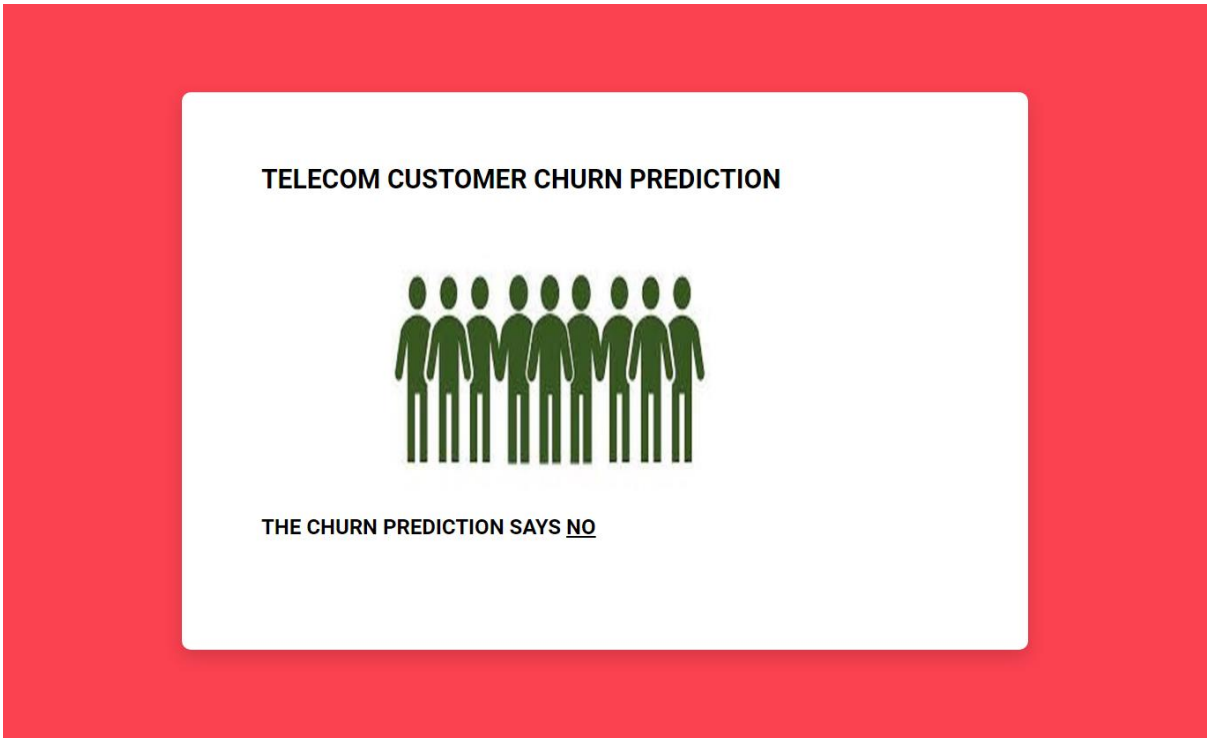
```
<style>
.center {
  margin: auto;
  width: 70%;
  padding: 10px;
}
</style>
<body>
    <div class="page-wrapper bg-red p-t-100 p-b-100 font-robo">
        <div class="wrapper wrapper--w960">
            <div class="card card-2">
                <div class="card-heading"></div>
                <div class="card-body">
                    <h2 class="title">Telecom Customer Churn Prediction</h2>
                    <div class="center">
                        <img src="../static/images/churn.jpg" width="400" height="300">
                    </div>
                    <h3 class="title"><p>The Churn Prediction says  <b><u>{{z}}</u></b></p></h3>


                </div>
            </div>
        </div>
    </div>

    <!-- Jquery JS-->
    <script src="{{ url_for('static', filename='vendor/jquery/jquery.min.js') }}" type="text/javascript"></script>
    <!-- Vendor JS-->
    <script src="{{ url_for('static', filename='vendor/select2/select2.min.js') }}" type="text/javascript"></script>


    <!-- Main JS-->
    <script src="{{ url_for('static', filename='js/global.js') }}" type="text/javascript"></script>

</body><!-- This templates was made by Colorlib (https://colorlib.com) -->

</html>
<!-- end document-->
```

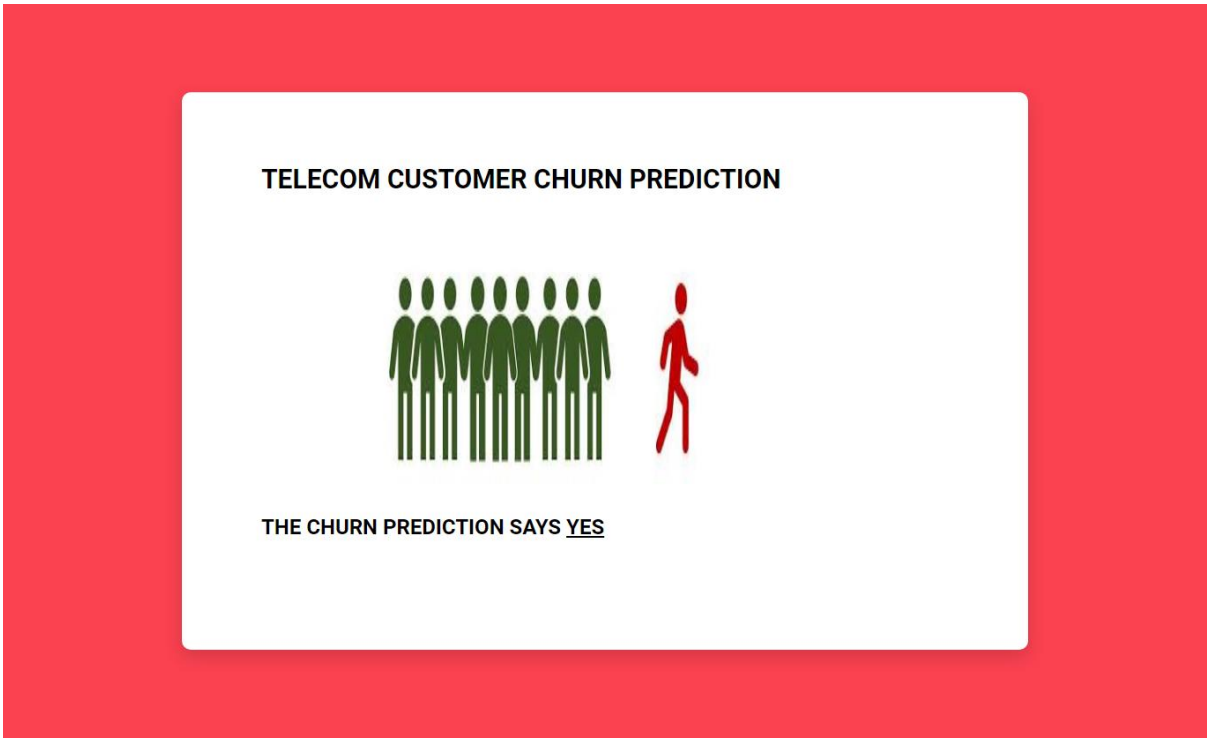
The html page looks like

Predbad.html

```
</head>
<style>
.center {
  margin: auto;
  width: 70%;
  padding: 10px;
}
</style>
<body>
    <div class="page-wrapper bg-red p-t-100 p-b-100 font-robo">
        <div class="wrapper wrapper--w960">
            <div class="card card-2">
                <div class="card-heading"></div>
                <div class="card-body">
                    <h2 class="title">Telecom Customer Churn Prediction</h2>
                    <div class="center">
                        <img src="../static/images/stays.jpg" width="400" height="300">
                    </div>
                    <h3 class="title"><p>The Churn prediction says  <b><u>{{z}}</u></b></p> </h3>
                </div>
            </div>
        </div>
    </div>

    <!-- Jquery JS-->
    <script src="{{ url_for('static', filename='vendor/jquery/jquery.min.js') }}" type="text/javascript"></script>
    <!-- Vendor JS-->
    <script src="{{ url_for('static', filename='vendor/select2/select2.min.js') }}" type="text/javascript"></script>


    <!-- Main JS-->
    <script src="{{ url_for('static', filename='js/global.js') }}" type="text/javascript"></script>

</body><!-- This templates was made by Colorlib (https://colorlib.com) -->

</html>
<!-- end document-->
```

The html page looks like





Activity 2: Main Python Script

Let us build app.py flask file which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application.

In order to develop web api with respect to our model, we basically use Flask framework which is written in python.

```python
from flask import Flask, render_template, request
app = Flask(__name__)
import pickle
model = pickle.load(open('churn.pkl','rb'))

@app.route('/')
def helloworld():
    return render_template("base.html")
@app.route('/assesment')
def prediction():
    return render_template("index.html")

@app.route('/predict', methods = ['POST'])
def admin():
    a= request.form["gender"]
    if (a == 'f'):
        a=0
    if (a == 'm'):
        a=1
    b= request.form["srcitizen"]
    if (b == 'n'):
        b=0
    if (b == 'y'):
        b=1
    c= request.form["partner"]
    if (c == 'n'):
        c=0
    if (c == 'y'):
        c=1
    d= request.form["dependents"]
    if (d == 'n'):
        d=0
    if (d == 'y'):
        d=1
    e= request.form["tenure"]
    f= request.form["phservices"]
    if (f == 'n'):
```

```
f= request.form["phservices"]
if (f == 'n'):
    f=0
if (f == 'y'):
    f=1
g= request.form["multi"]
if (g == 'n'):
    g1,g2,g3=1,0,0
if (g == 'nps'):
    g1,g2,g3=0,1,0
if (g == 'y'):
    g1,g2,g3=0,0,1
h= request.form["is"]
if (h == 'dsl'):
    h1,h2,h3=1,0,0
if (h == 'fo'):
    h1,h2,h3=0,1,0
if (h == 'n'):
    h1,h2,h3=0,0,1
i= request.form["os"]
if (i == 'n'):
    i1,i2,i3=1,0,0
if (i == 'nis'):
    i1,i2,i3=0,1,0
if (i == 'y'):
    i1,i2,i3=0,0,1
j= request.form["ob"]
if (j == 'n'):
    j1,j2,j3=1,0,0
if (j == 'nis'):
    j1,j2,j3=0,1,0
if (j == 'y'):
    j1,j2,j3=0,0,1
k= request.form["dp"]
if (k == 'n'):
    k1,k2,k3=1,0,0
if (k == 'nis'):
    k1,k2,k3=0,1,0
if (k == 'nis'):
    k1,k2,k3=0,1,0
if (k == 'y'):
    k1,k2,k3=0,0,1
l= request.form["ts"]
if (l == 'n'):
    l1,l2,l3=1,0,0
if (l == 'nis'):
    l1,l2,l3=0,1,0
if (l == 'y'):
    l1,l2,l3=0,0,1
m= request.form["stv"]
if (m == 'n'):
    m1,m2,m3=1,0,0
if (m == 'nis'):
    m1,m2,m3=0,1,0
if (m == 'y'):
    m1,m2,m3=0,0,1
n= request.form["smv"]
if (n == 'n'):
    n1,n2,n3=1,0,0
if (n == 'nis'):
    n1,n2,n3=0,1,0
if (n == 'y'):
    n1,n2,n3=0,0,1
o= request.form["contract"]
if (o == 'mtm'):
    o1,o2,o3=1,0,0
if (o == 'oyr'):
    o1,o2,o3=0,1,0
if (o == 'tyrs'):
    o1,o2,o3=0,0,1
p= request.form["pmt"]
if (p == 'ec'):
    p1,p2,p3,p4=1,0,0,0
```

```
    p= request.form["pmt"]
    if (p == 'ec'):
        p1,p2,p3,p4=1,0,0,0
    if (p == 'mail'):
        p1,p2,p3,p4=0,1,0,0
    if (p == 'bt'):
        p1,p2,p3,p4=0,0,1,0
    if (p == 'cc'):
        p1,p2,p3,p4=0,0,0,1
    q= request.form["plb"]
    if (q == 'n'):
        q=0
    if (q == 'y'):
        q=1
    r= request.form["mcharges"]
    s= request.form["tcharges"]

    t=[[int(g1),int(g2),int(g3),int(h1),int(h2),int(h3),int(i1),int(i2),int(i3),i
    x = model.predict(t)
    if (x[0] == 0):
        y ="No"
        return render_template("predno.html", z = y)

    if (x[0] == 1):
        y ="Yes"
        return render_template("predyes.html", z = y)

if __name__ == '__main__':
    app.run(debug = True)
```

Activity 3: Run the App

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page

```
(tf) C:\Users\ACER\Desktop\data science\Telecom churn modelling\flask app>python
 app.py
```

Then it will run on local host:5000

```
(tf) C:\Users\ACER\Desktop\data science\Telecom churn modelling\flask app>python
 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 269-632-688
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- **Activity 5:**

Output Screenshots:

TELECOM CUSTOMER CHURN PREDICTION

Customer churn has become highly important for companies because of increasing competition among companies, increased importance of marketing strategies and conscious behaviour of customers in the recent years. Customers can easily trend toward alternative services. Companies must develop various strategies to prevent these possible trends, depending on the services they provide. During the estimation of possible churns, data from the previous churns might be used. An efficient churn predictive model benefits companies in many ways. Early identification of customers likely to leave may help to build cost effective ways in marketing strategies. Customer retention campaigns might be limited to selected customers but it should cover most of the customer. Incorrect predictions could result in a company losing profits because of the discounts offered to continuous subscribers.

Click me to continue with prediction

PREDICTION FORM

Gender	Yes
Yes	Yes
3	Yes
No Phone service	DSL
No	Yes
No	No
Yes	Yes
Month to Month	Yes
Bank Transfer(Automatic)	39.5
39.5	

Submit

TELECOM CUSTOMER CHURN PREDICTION

THE CHURN PREDICTION SAYS NO

TELECOM CUSTOMER CHURN PREDICTION

THE CHURN PREDICTION SAYS YES