

Customer Churn Prediction (Telecom Industry)

**ABSTRACT :-**

Churn is a one of the biggest problem in the telecom industry. Research has shown that the average monthly churn rate among the top 4 wireless carriers in the US is 1.9% - 2%. A common problem faced by most Telecom industries is to retain their customer base. We are working here to identify all such potential users based on some past customer's behavior that how likely they can stop using services. Eventually, this will help us to work on those features which are causing major dropouts so that we can work on those areas. We will do this by building predictive models and based on their results, we will be able to classify the customer type i.e. "CHURN" and "NOT CHURN". The above-mentioned problem can be nullified to some extent by the proposed system. The execution is performed with the help of data acquisition followed by data cleaning, performing statistical methods, data modeling, Evaluation of model, Deployment through Heroku, Flask API and user interface.

**SOURCES :-**



Fig : Data Source

**PROBLEM STATEMENT:-**

Based on the introduction the key challenge is to predict if an individual customer will churn or not. To accomplish that, machine learning models are trained based on 80% of the sample data. The remaining 20% are used to apply the trained models and assess their predictive power with regards to “churn / not churn”.

To compare models and select the best for this task, the accuracy is measured. Based on other characteristics of the data, for example the balance between classes (number of “churners” vs. “non-churners” in data set) further metrics is considered if needed.

**OBJECTIVES :-**

* Which is the most important factor that contributes to the high retention rate?
* Which analytics model can accurately predict a customer’s churn rate?
* Deploy the machine learning model using Flask API and pickle files.
* Design a UI for entering model inputs and display results accordingly.

**PROJECT FLOW :-**

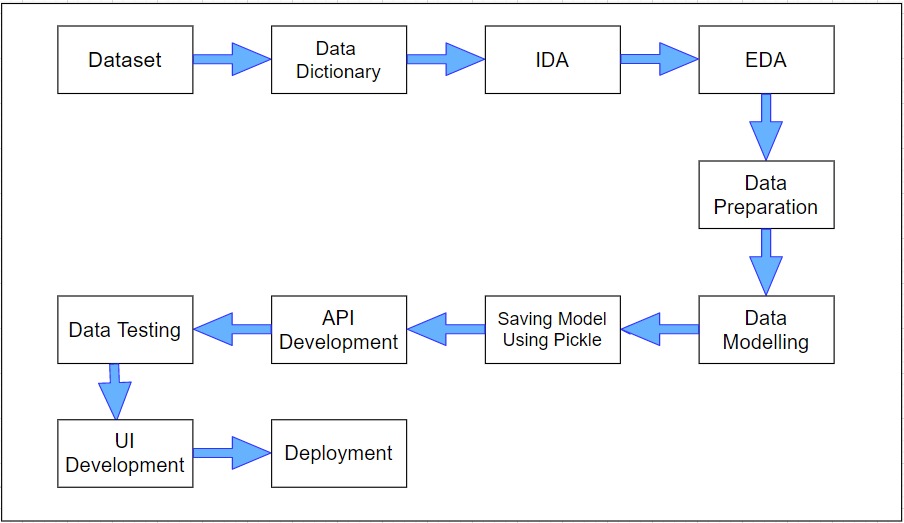
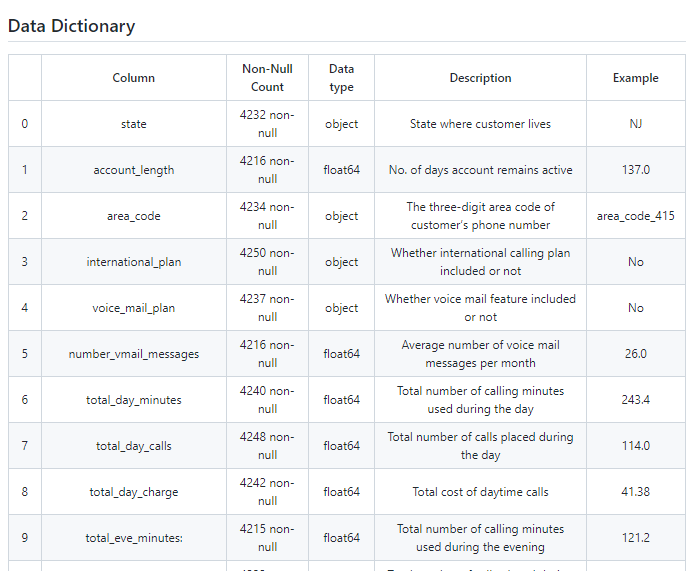
****

Fig.: Overall flow of project

The detail working of each flow is further explained below:-

* **DATASET : -** The Telco Company’s data set is available [**on Kaggle**](https://www.kaggle.com/becksddf/churn-in-telecoms-dataset). The company provides mobile network services to 4250 customers in overall United States. Our challenge is to help the company predict behavior to retain customers and analyze all relevant customer data to develop focused customer retention programs.
* **DATA DICTIONARY : -** This data dictionary tells about the features name and their short information of what they belong to.



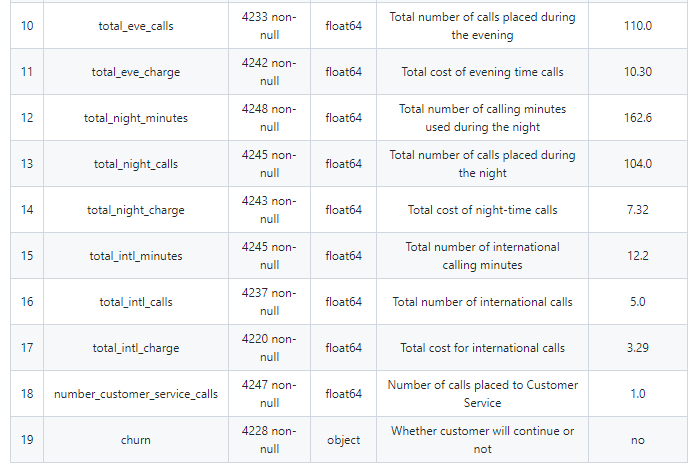
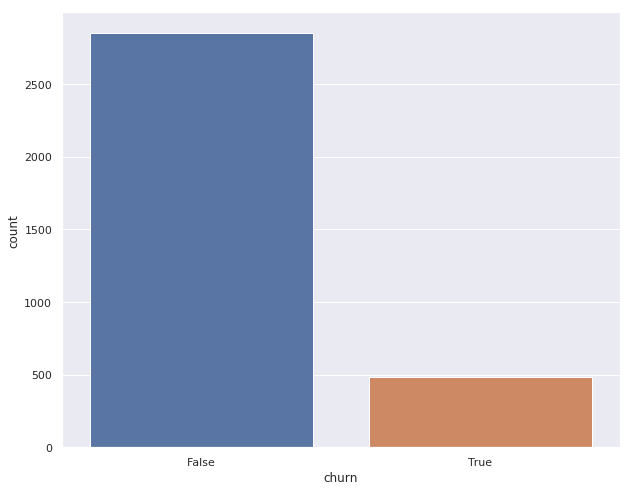
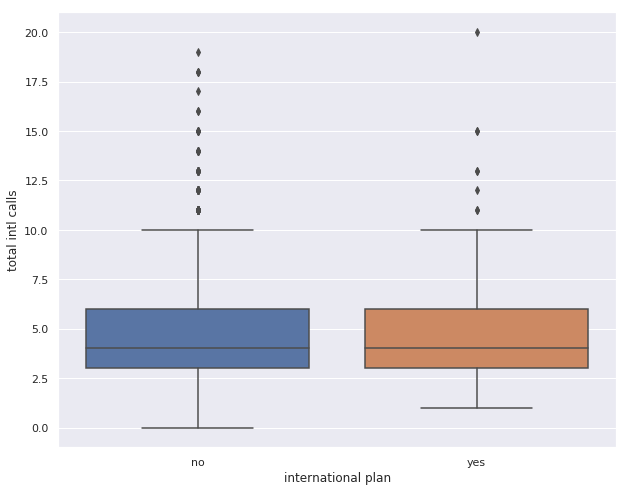
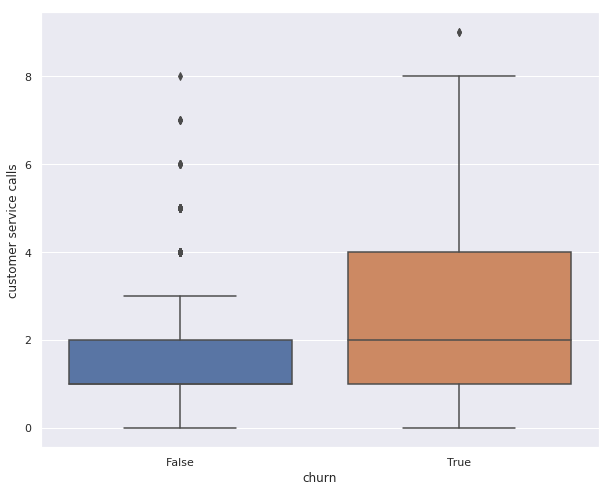


Fig. 3.3 Data Dictionary of the Dataset

* **IDA : -** We have performed the Initial Data Analysis which demonstrates the overview Data Visualization of the dataset. Some of them are.



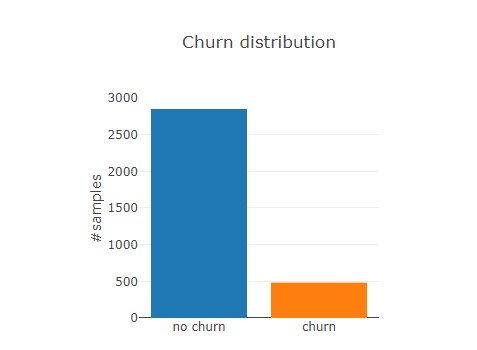




As see from the above figures, there are almost 85% of the customers who are not churned that means, most of the customers are using our services. But, what about the rest 15%? When we come down we also analyzed the impact of international calls and also how many times the customer has made service calls. The more no. of international calls suggest about premium customers and high rate of calls to customer services means they are facing some issues with the network. There is more analysis drilled down in the EDA part.

* **EXPLORATORY DATA ANALYSIS : -**

We did the EDA on Jupyter notebook using visualization libraries like Matplotlib, pyplot and Seaborn. Some of the analysis is stated below.

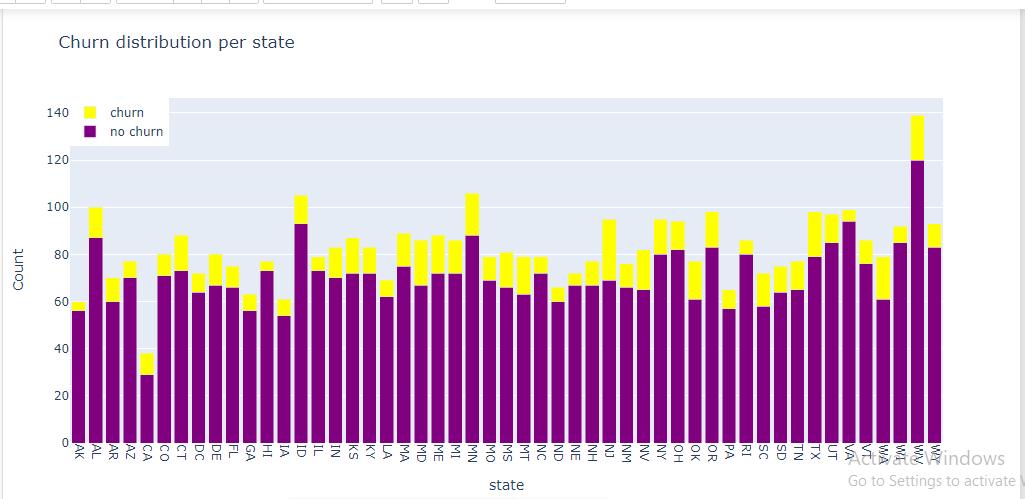




We can see that we have clearly more samples for customers without churn than for customers with churn. So we have a class imbalance for the target variable which could lead to predictive models which are biased towards the majority (i.e. no churn).

In order to deal with this issue we will investigate into the use of oversampling when building the models.

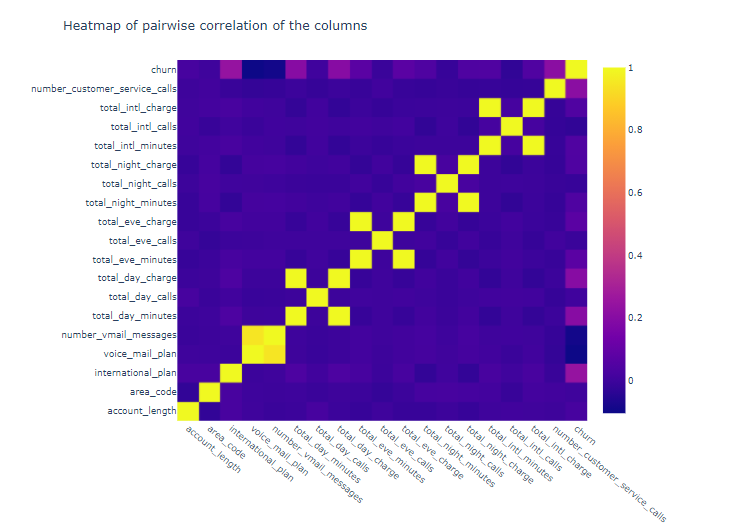
State-wise Churn Distribution



We can see that some states have less proportion of customer with churn like AK, HI, IA and some have a higher proportion such as WA, MD and NJ. This shows that we should incorporate the state into our further analysis, because it could be help to predict if a customer is going to churn.

As it can been from above diagram that the most churned customer belonged to NJ (New Jersey) with 26 customers followed by WV (West Virginia) 19 customers and so on…

Co-relation Matrix



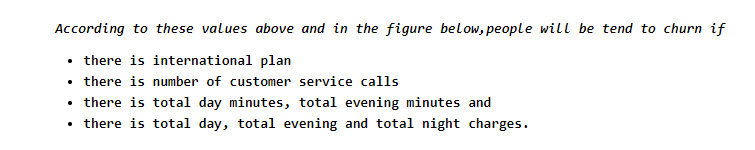
There were features such as ‘voice mail plan’, ‘voice mail messages’, ‘total day charge’, ‘total day minutes’, ‘total eve charge’, ‘total eve minutes’, ‘total night charge’, ‘total night minutes’, ‘total intl charges’, ‘total intl minutes’ which were having more co-relation in them. That could be obvious because as these the more no. of minutes the greater is the amount of charge. The same behavior can be seen for the evening and the night calls.

We can see a high correlation between the voice mail plan and the number of voice mail messages. It makes sense that customers with the voice mail plan also send more voice mail messages.

However, the international plan is just slightly correlated with the total international minutes and the international charge.

The highest correlation with the churn variable has the international plan, the total\_day\_charge, the total\_day\_minutes and the number of customer service calls.

Also, some other features were more contribution towards the target which will be discussed later in the book. Some more conclusions are listed below.



* **DATA PREPARATION :-**

To make the data ready for model building, we need to organize the data in the standard format. The steps which we followed while preparing the data are:

1. Encoding the categorical variables with different encoding techniques which included Label Encoding and Category encoding.
2. Handling missing values with statistical method approach. Here, all the numerical missing values were handled through median of each feature rather than taking mean as there were outliers in most of the missing columns. Likewise, all the categorical missing values were handled by replacing it with the most frequent occurrence category of that column.
3. As there were some outliers in our data, we tried to first identify the outliers through the 1.5 IQR rule. By doing this, we found that there were some outliers which were again handled through the median technique for each column.

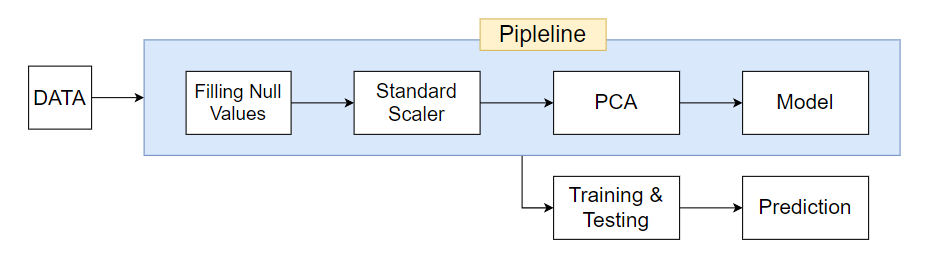
These are some of the steps we followed for data preparation.

* **DATA MODELING :-**

Data modeling is the process of creating a visual representation of either a whole information system or parts of it to communicate connections between data points and structures. ... This provides a common, consistent, and predictable way of defining and managing data resources across an organization, or even beyond.

The steps which we followed while modeling the data are:

1. We used 80% - 20% train-test split of data.
2. We created model pipelines to exercise control over the ML model. A well-organized pipeline makes the implementation more flexible. It is like having an exploded view of a computer where you can pick the faulty pieces and replace them- in our case, replacing a chunk of code. Here We will define our pipeline in three stages:



* We had use *SimpleImputer* to Imputing missing values in the categorical column using the most frequent and numerical column using the median.
* We used a ColumnTransformer to do the required transformations it contains Scaling the required column  using  StandardScaler().
* Build different models to predict the target with best model.

1. **Using Scikit Libraries:**
2. Logistic Regression
3. Random Forest
4. Support Vector Machine (svm)

**Parameter Settings for Sci-kit Packages:** We tried out various parameter tuning settings to find out the best ones, based on a greedy approach, i.e. changing one parameter at a time, for each of the algorithm on each of the data set separately. You may see the output of these parameter setting by executing file provided. Below are the best parameter settings for each of the algorithms on each dataset:

1. **Logistic Regression**

Accuracy of our model with Confusion matrix :-

***Confusion Matrices:***

|  |  |  |
| --- | --- | --- |
| ***Actual Classss***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *724* | *0* |
| ***1*** | *123* | *3* |
| Test Accuracy : **0.85**  True Positive Rate :  **1.0**  False Positive Rate : **0.97** | | |

**After hyper tuning the model**

|  |  |  |
| --- | --- | --- |
| ***Actual Classss***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *724* | *0* |
| ***1*** | *119* | *7* |
| Test Accuracy : **0.86**  True Positive Rate :  **1.0**  False Positive Rate : **0.94** | | |

**After Thresholding**

|  |  |  |
| --- | --- | --- |
| ***Actual Class***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *699* | *25* |
| ***1*** | *68* | *58* |
| Test Accuracy : **0.89**  True Positive Rate :  **0.96**  False Positive Rate : **0.53** | | |

1. **Random Forest**

Accuracy of our model with Confusion matrix :-

***Confusion Matrices:***

|  |  |  |
| --- | --- | --- |
| ***Actual Class***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *719* | *5* |
| ***1*** | *99* | *27* |
| Test Accuracy : **0.88**  True Positive Rate :  **0.99**  False Positive Rate : **0.76** | | |

**After hyper tuning the model**

|  |  |  |
| --- | --- | --- |
| ***Actual Class***  ***class***  ***cl***  ***Actual Classss***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *717* | *7* |
| ***1*** | *97* | *29* |
| Accuracy : **0.88**  True Positive Rate :  **0.99**  False Positive Rate : **0.67** | | |

**After Thresholding**

|  |  |  |
| --- | --- | --- |
| ***Actual Class***  ***class***  ***cl***  ***Actual Classss***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *668* | *56* |
| ***1*** | *59* | *67* |
| Accuracy : **0.88**  True Positive Rate :  **0.92**  False Positive Rate : **0.46** | | |

1. **Support Vector Machine**

Accuracy of our model with Confusion matrix :-

***Confusion Matrices:***

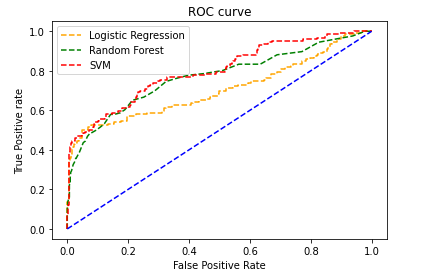
|  |  |  |
| --- | --- | --- |
| ***Actual Class***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *720* | *4* |
| ***1*** | *88* | *38* |
| Test Accuracy : **0.89**  True Positive Rate :  **0.99**  False Positive Rate : **0.69** | | |

**After hyper tuning the model**

|  |  |  |
| --- | --- | --- |
| ***Actual Class***  ***class***  ***cl***  ***Actual Classss***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *724* | *0* |
| ***1*** | *86* | *40* |
| Accuracy : **0.89**  True Positive Rate :  **1.0**  False Positive Rate : **0.68** | | |

**After Thresholding**

|  |  |  |
| --- | --- | --- |
| ***Actual Class***  ***class***  ***cl***  ***Actual Classss***  ***Predicted***  ***Class*** | ***0*** | ***1*** |
| ***0*** | *717* | *7* |
| ***1*** | *71* | *55* |
| Accuracy : **0.90**  True Positive Rate :  **0.99**  False Positive Rate : **0.56** | | |

****

* **PICKLING :-**

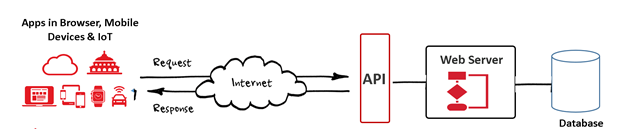
Pickle is a module in Python used for serializing and de-serializing Python objects. This converts Python objects like lists, dictionaries, etc. into byte streams (zeroes and ones). You can convert the byte streams back into Python objects through a process called unpickling. Pickling is also known as serialization, flattening, or marshalling.

The pickle.load(what we want to load) will use the .pickle file by loading it in the memory and then we can use the model for testing and creating web app easily.

* **API DEVELOPMENT :-**

API stands for Application Programming Interface. In basic terms, APIs are a set of functions and procedures that allow for the creation of applications that access data and features of other applications, services, or operating systems.

Good APIs make it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer.



Flask is a web framework for python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates. For API, we required to import flask, jsonify. Jsonify is a function in Flask's flask.json module. Jsonify serializes data JSON format. The two most common requests are GET, which pulls data from a server and POST, which pushes data from a server. After running the code, we can see an http link, we copied this link and gave to a postman (is an API and development tool which helps to build, test and modify APIs. It has the ability to make various types of HTTP requests, saving environments for later use, converting the API to code for various languages like JavaScript, python).

* **DATE TESTING :-**

It is the input given to a program during test execution. It represents data that affects or affected by software execution while testing. We have created report for test results in which the number of, expected results, actual results and accuracy. It gave us 85-90% good results and 5-8% bad results. So, we can conclude that data testing is successful.

Here the Classification Report of our models:

**Classification Report for Logistic Regression :**

Precision recall f1-score support

0 0.91 0.97 0.94 724

1 0.70 0.46 0.56 126

**accuracy 0.89**  850

macro avg 0.81 0.71 0.75 850

weighted avg 0.88 0.89 0.88 850

**Classification Report for Random Forest :**

precision recall f1-score support

0 0.91 0.95 0.93 724

1 0.62 0.48 0.54 126

**accuracy 0.88** 850

macro avg 0.77 0.72 0.74 850

weighted avg 0.87 0.88 0.87 850

**Classification Report for Support Vector**

**Machine :**

precision recall f1-score support

0 0.91 0.99 0.95 724

1 0.89 0.43 0.58 126

**accuracy 0.91** 850

macro avg 0.90 0.71 0.76 850

weighted avg 0.91 0.91 0.89 850

* **UI DEVELOPMENT :-**

In front-end we have created a website with the help of HTML, CSS and bootstrap framework. Bootstrap is a powerful toolkit - a collection of HTML, CSS, and JavaScript tools for creating and building web pages and web applications. In Back-end, we have used flask framework to build web-application. It fetch the data from our python(py) file to website. As a final result we can see a website with different tabs User can compare customer

Activity and accordingly make decision. Also there is a button provided like predict, by clicking on it user will automatically get redirected to the final result link and can see the prediction.

* **DEPLOYMENT:-**

We have created a wire-frame design using the Heroku. Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku **to deploy, manage, and scale modern apps**. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market. In back-end we have used flask API which is used for managing http requests. Here we used fetch API function to get the data. And used this data to create a front-end. In front-end we have created a website with the help of HTML, CSS and bootstrap framework. Bootstrap is a powerful toolkit - a collection of HTML, CSS, and JavaScript tools for creating and building web pages and web applications. As a final result we can see a website with different tabs with customer activity and a button provided like predict, by clicking on it user will automatically get redirected to the decision link and can see the final prediction.

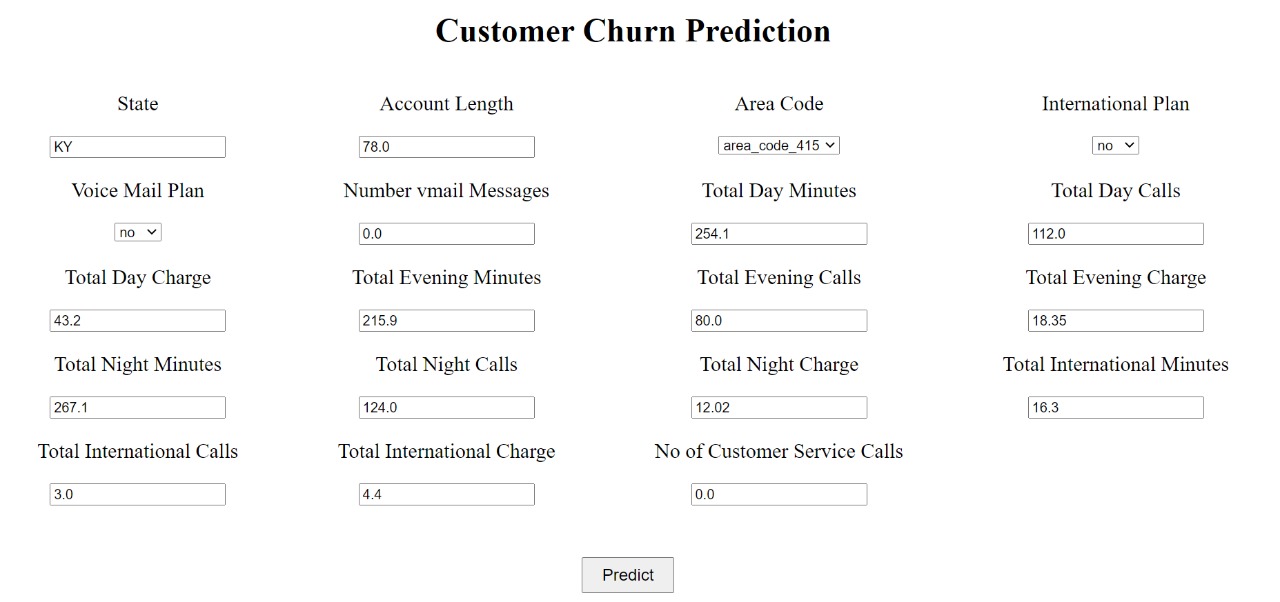
****

Fig : Web App

# ****Technology used****

| **Element** | **Description** |
| --- | --- |
| **Python** | Programming language |
| **Numpy** | Python library used for advanced Mathematical operations |
| **Pandas** | Python library used for data manipulation & analysis |
| **Matplotlib and Seaborn** | Python library used for data visualization |
| **Sklearn** | Python library used for data pre-processing and modeling |
| **pickle** | Python library used for saving model |
| **Requests** | Python library used for making HTTP requests |
| **Flask** | Used for making web applications and creating API |
| **HTML CSS** | Basic language for website designing |
| **Bootstrap** | CSS framework used to create interactive website |
| **Heroku** | Cloud Platform used for deploying data models |

# Conclusion:-

* We have stored data in a CSV file to consume later which can be used to create data files.
* We had to perform lot of data cleanup and data mapping which helps us in building and maintain datasets.
* We were able to handling missing values by using most frequent and middle values.
* We were able to use pipeline to manage the workflow of our models.
* We built different models to predict the target and get the best model on the basis of their accuracy.
* Finally we built a system by which we were able to make prediction.

# Future Scope:-

* We can try and answer a lot of questions for which data is present online we can extract data, classify data, clean the data and then perform logical calculations to get more articulate data and try to answer a specific question.
* We can keep tabs of metrics and update them daily wherein we would have to get the data from sources clean the data and update it to the dataset regularly with a script run regularly.
* We can create a decision system offering the service of comparing multiple services when scaled and add more features when we can try to scale for result.
* We can add data related to other domains and use this system to take informed decisions. This helps company to retain their customer and increase the revenue.