PROJECT PHASE 3 REPORT

Section: CSE 587 B

Revathi Gollapudi -- vgollapu@buffalo.edu

Srikar Chintha -- chintha@buffalo.edu

Sumana Madhireddy – sumanama@buffalo.edu

Forecasting Telecom Customer Churn for Proactive Retention and Business Success.

Problem Statement

The telecommunications industry is dealing with a prevalent issue of customer churn, where subscribers discontinue services, leading to revenue loss and market share erosion. The problem at hand is to develop an effective predictive model for telecom customer churn and subsequently implement strategic retention measures.

Abstract

Customer churn prediction is a method that forecasts when consumers are likely to leave a service or subscription for a variety of reasons in industries such as finance, telecommunications, and so on. Customer retention has become increasingly important for businesses due to their constantly changing needs. In order to solve this problem, we developed a Customer Churn Prediction model with a Random Forest classifier and integrated it into "Customer Churn Predictor" app that we developed using Flask. Based on a few variables, the predictor app forecasts whether a consumer in the telecom sector will cancel their service or not.

Dataset

Dataset has been taken from the sample dataset from IBM sample dataset for Telecom Customer Churn.

https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=samples-telcocustomer-churn

The dataset has 7113 rows and 22 columns.

Below are the column names which are features for further analysis and their datatypes from the above dataset. Out of which 4 columns are float type, 1 column is of integer type and remaining 17 are Object type.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7113 entries, 0 to 7112
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7102 non-null   object 
 1   gender            7073 non-null   object 
 2   SeniorCitizen     7102 non-null   float64
 3   Partner           7102 non-null   object 
 4   Dependents        7070 non-null   object 
 5   tenure            7102 non-null   float64
 6   PhoneService      7102 non-null   object 
 7   MultipleLines     7102 non-null   object 
 8   InternetService   7102 non-null   object 
 9   OnlineSecurity    7102 non-null   object 
 10  OnlineBackup      7102 non-null   object 
 11  DeviceProtection  7102 non-null   object 
 12  TechSupport       7102 non-null   object 
 13  StreamingTV       7102 non-null   object 
 14  StreamingMovies   7102 non-null   object 
 15  Contract          7058 non-null   object 
 16  PaperlessBilling  7056 non-null   object 
 17  PaymentMethod     7102 non-null   object 
 18  MonthlyCharges    7053 non-null   float64
 19  TotalCharges      7102 non-null   float64
 20  Churn             7102 non-null   object 
 21  Count             7113 non-null   int64  
dtypes: float64(4), int64(1), object(17)
memory usage: 1.2+ MB
```

Model

In our project we have implemented K-nearest Neighbors, Logistic Regression, Support Vector Machine Classifier, Naïve Bayes, Random Forest, and Gradient Boosting. Among all of them we choose Random Forest classifier.

Random Forest had the highest accuracy (79.22%), followed by Logistic Regression (78.05%) and Gradient Boosting (78.73%). Additionally, Random Forest had the strongest recall (49.03%), demonstrating how well it captures real positive experiences. Since SVM and Naive Bayes had zero recall, it is possible that they were unable to detect positive examples. The best precision was achieved using logistic regression (66.03%), which was closely followed by gradient boosting (66.67%) and random forest (66.99%).

Random Forest is also well known for its great predicting accuracy. It is also an ensemble learning technique that yields predictions by combining many decision trees. Several trees work together to increase generalization performance and minimize overfitting.

Random Forest is also an ensemble model which combines the predictions of base models which in case of random forests are decision trees.

Random Forest lowers the chance of overfitting by combining predictions from several decision trees with above medium size dataset of 7113 rows and 22 columns. Each tree's depth and complexity are adjustable, with this we can customize our model to get the best accuracy by testing on different values.

Below is the implementation of Random Forest classifier Using scikit-learn

5. Random Forest Algorithm

```
import pickle
# Create a Random Forest classifier with a specified 100 trees (n_estimators)
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42,max_depth=12)

# Fit the classifier on the training data
rf_classifier.fit(X_train, Y_train.ravel())

# Make predictions on the test data
y_pred_rf = rf_classifier.predict(X_test)


#Calculated Accuracy, Recall, Precision, F1_score for the above predictions
print("Accuracy =",Accuracy(Y_test,y_pred_rf))
print("Recall =",recall(Y_test,y_pred_rf))
print("Precision =",precision(Y_test,y_pred_rf))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,y_pred_rf)
plot_roc_curve(Y_test,y_pred_rf)
```

Effectives of the Algorithm on chosen dataset:

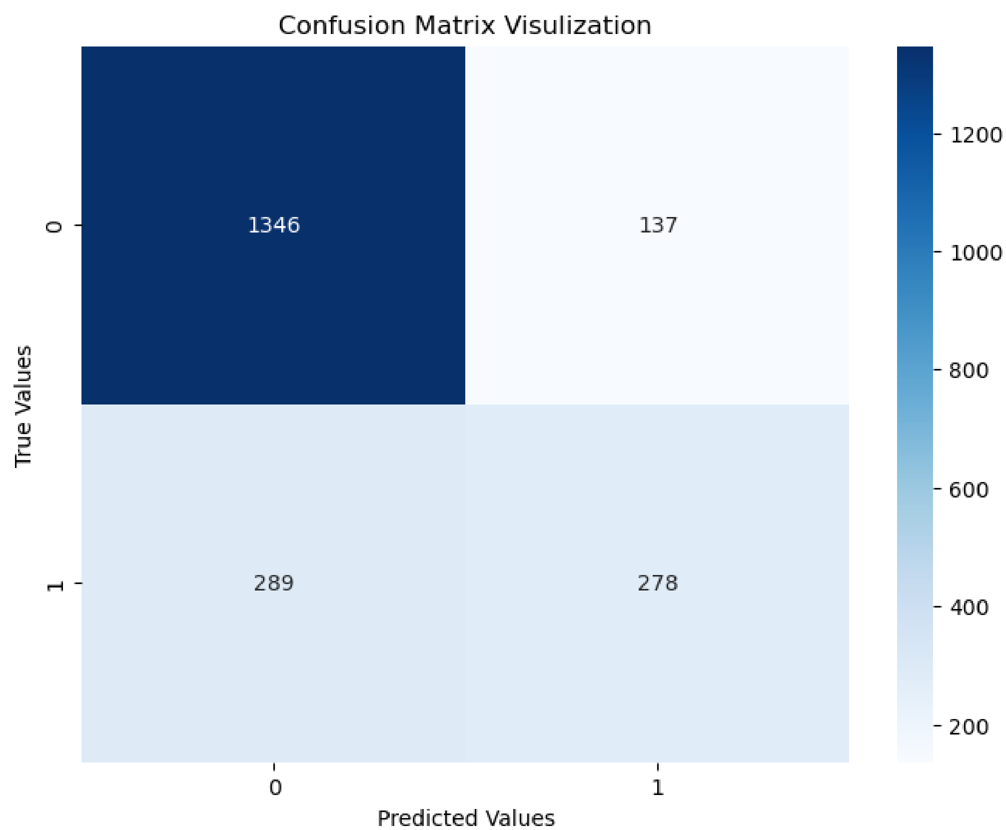
The model exhibits an accuracy of about 79.2%, indicating a relatively high rate of overall correct classifications. However, the recall is just over 49.02%, suggesting it correctly identifies only half of the positive cases, while the precision is higher at 67%, indicating a better rate of predicting true positives out of all positive predictions.

```
Accuracy = 79.21951219512195
Recall = 0.4902998236331569
Precision = 0.6698795180722892
F1 Score = 0.4884488448844885
```

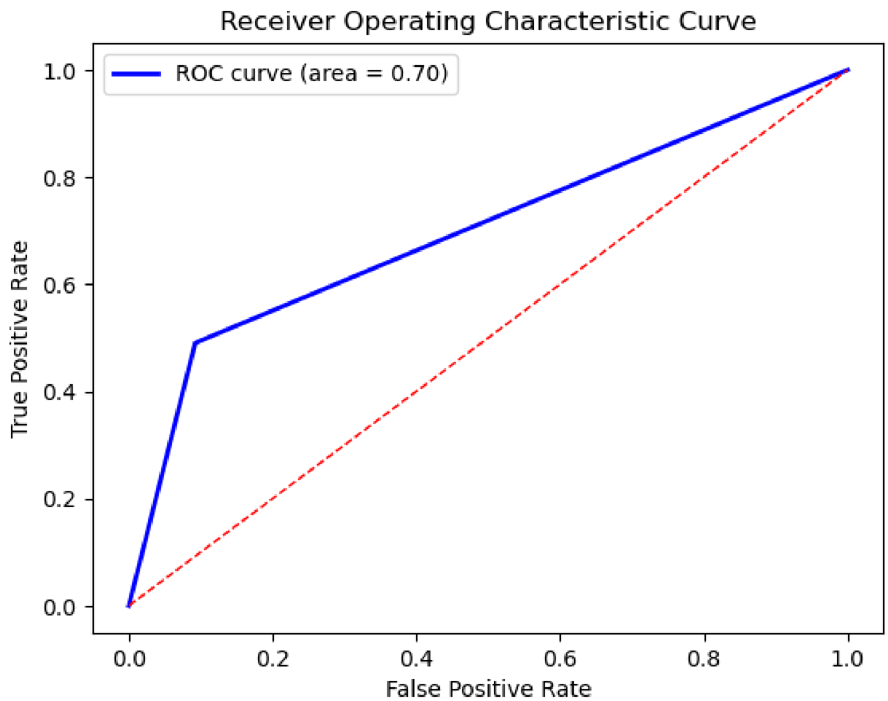
We have saved our model as “best_model_rf” as a pickle file which we have used in our flask program.

```python
pickle.dump(rf_classifier, open( "best_model_rf.p", 'wb' ))
```

The confusion matrix below indicates the model's strong ability to correctly predict the negative class with 1346 true negatives, but it also shows a significant number of false negatives, with 289 instances where the positive class was incorrectly predicted as negative. Conversely, the model predicted 278 true positives, indicating moderate effectiveness in identifying the positive class, and made 137 false positives, where the negative class was incorrectly predicted as positive.



An AUC of 0.70 for a Random Forest classifier tells that it can differentiate between positive and negative classes with good accuracy, yet there's potential for enhancing its predictive performance.



Deployment:

In our HTML script, we have used a form which takes input and predicts if customer is going to stay or leave as churn Yes or No. We also used a “predict churn” button to output the prediction.

Input parameters the form includes are Gender, Monthly Charges, Payment Method, Paperless Billing, Senior Citizen, Phone Service, Multiple Lines, Total Charges.

The action property on the form instructs the POST method to be used to send the data to the '/predict' site. The projected churn value is shown in the 'predicted_value' div after the user's input has been analyzed by the server. The code also uses Flask's 'url_for' function to insert an image for a particular photo. Input fields, dropdown menus, and labels are used in the form's layout to capture key information needed to anticipate customer churn.

When the user clicks the "View Visualization Analysis" button, the JavaScript code within the script element manages the dynamic loading of images, enriching the user experience with extra visual insights connected to the dataset.

The script is as shown below. HTML script is placed in templates folder and CSS script is placed in static folder respectively.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Churn Predictor</title>
    <link rel="stylesheet" href="{{ url_for('static', filename = 'css/styles.css')}}">
</head>
```

```html
<body>
  <div class = 'main_line'>
    <div class="header">
      <img src="{{ url_for('static',filename = 'css/churn_photo.PNG')}}" alt="Churn Photo" class="churn-photo">
      <h1>Enter below details to predict Customer Churn:</h1>
    </div>
    <form  action="/predict" method="post">
        <label for="gender">Gender:</label>
        <select id="gender" name="gender" required>
            <option value="" disabled selected>Select Gender</option>
            <option value="male">Male</option>
            <option value="female">Female</option>
        </select><br>
        <label for="MonthlyCharges">Monthly Charges:</label>
        <input type="text" id="MonthlyCharges" name="MonthlyCharges" class="centered-input "required><br>
        <label for="PaymentMethod">Payment Method:</label>
        <select id="PaymentMethod" name="PaymentMethod" required>
            <option value="" disabled selected>Select Payment Method</option>
            <option value="Bank transfer">Bank Transfer</option>
            <option value="Credit card">Credit Card</option>
            <option value="Electronic check">Electronic Check</option>
            <option value="Mailed check">Mailed Check</option>
        </select><br>
        <label for="PaperlessBilling">Paperless Billing:</label>
        <select id="PaperlessBilling" name="PaperlessBilling" required>
            <option value="" disabled selected>Select Paperless Billing Option</option>
            <option value="yes">Yes</option>
            <option value="no">No</option>
        </select><br>
        <label for="SeniorCitizen">Senior Citizen:</label>
        <select id="SeniorCitizen" name="SeniorCitizen" required>
            <option value="" disabled selected>Select Senior Citizen Option</option>
            <option value="yes">Yes</option>
            <option value="no">No</option>
        </select><br>
```

```html
        <label for="PhoneService">Phone Service:</label>
        <select id="PhoneService" name="PhoneService" required>
            <option value="" disabled selected>Select Phone Service Option</option>
            <option value="yes">Yes</option>
            <option value="no">No</option>
        </select><br>
        <label for="MultipleLines">Multiple Lines:</label>
        <select id="MultipleLines" name="MultipleLines" required>
            <option value="" disabled selected>Select Multiple Lines Option</option>
            <option value="No">No</option>
            <option value="Yes">Yes</option>
        </select><br>
        <label for="TotalCharges">Total Charges:</label>
        <input type="text" id="TotalCharges" name="TotalCharges" class="centered-input" required><br><br>
        <button type="submit">Predict Churn</button>
        <button type="button" id="viewImagesBtn">View Visualization Analysis of the Churn Prediction Dataset</button>
    </form>
    <div id="imageContainer"></div>
    <script>
        document.getElementById('viewImagesBtn').addEventListener('click', function(event) {
            var imageUrls = ['{{ url_for("static", filename="css/senior_citizen.PNG") }}',
                             '{{ url_for("static", filename="css/gender.PNG") }}',
                             '{{ url_for("static", filename="css/payment_method.PNG") }}'];
            imageUrls.forEach(function(url) {
                var img = document.createElement('img');
                img.src = url;
                img.style.width = '500px';
                img.style.height = 'auto';
                document.getElementById('imageContainer').appendChild(img);
            });
        });
    </script>
    <br>
    <div id='predicted_value'>{{ predicted_value }}</div>
  </div>
</body>
</html>
```

Additionally, to further format our front-end application for deployment, we have employed a CSS script. In addition to improving the appearance, we have added an image.

```css
body {
    margin: 0;
    padding: 0;
    font-family: "Arial", sans-serif;
}
.main_line {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    background-color: lightyellow;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
.header {
    text-align: center;
    margin-bottom: 20px;
}
.churn-photo {
    width: 20vw;
    height: auto;
    border-radius: 0px;
    object-fit: fill;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    background-color: lightyellow;
}
.centered-input {
    width: 10%;
    padding: 8px;
    margin: 10px auto;
    box-sizing: border-box;
    border: 1px solid #ddd;
    border-radius: 4px;
    display: inline-block;
}
@keyframes backgroundSlide {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}
form {
  text-align: center;
  background-color: #f0f0f0;
  background-image: linear-gradient(45deg, #6D9B98, #E0CA3C, #E37B40, #6D9B98);
  background-size: 400% 400%;
  animation: backgroundSlide 15s ease infinite;
}
form:hover {
  background-color: #e0e0e0;
  transition: background-color 0.3s ease;
}
label {
  color: #000;
  padding: 5px 10px;
  border-radius: 5px;
  font-weight: bold;
  display: inline-block;
  margin-bottom: 5px;
}
.display-image {
    width: 100%;
    height: auto;
    margin: 10px 0;
}
input {
    width: calc(100% - 16px);
    padding: 8px;
    margin-bottom: 16px;
    box-sizing: border-box;
    border: 1px solid #ddd;
    border-radius: 4px;
}
button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
button:hover {
    background-color: #45a049;
}
#predicted_value {
    margin-top: 20px;
    text-align: center;
    font-weight: bold;
    color: #333;
}
```

This Python script creates a Flask web application that uses a machine learning model that has been built to forecast the customer churn.

The application uses a pickled file to load a Random Forest model that has already been trained, and it integrates the Flask web framework to generate an interface.

Through an HTML form, users provide information about Gender, Monthly Charges, Payment Method, Paperless Billing, Senior Citizen, Phone Service, Multiple Lines, Total Charges.

Subsequently, the preprocessed input data is incorporated into the model to forecast the likelihood of a customer's retention.

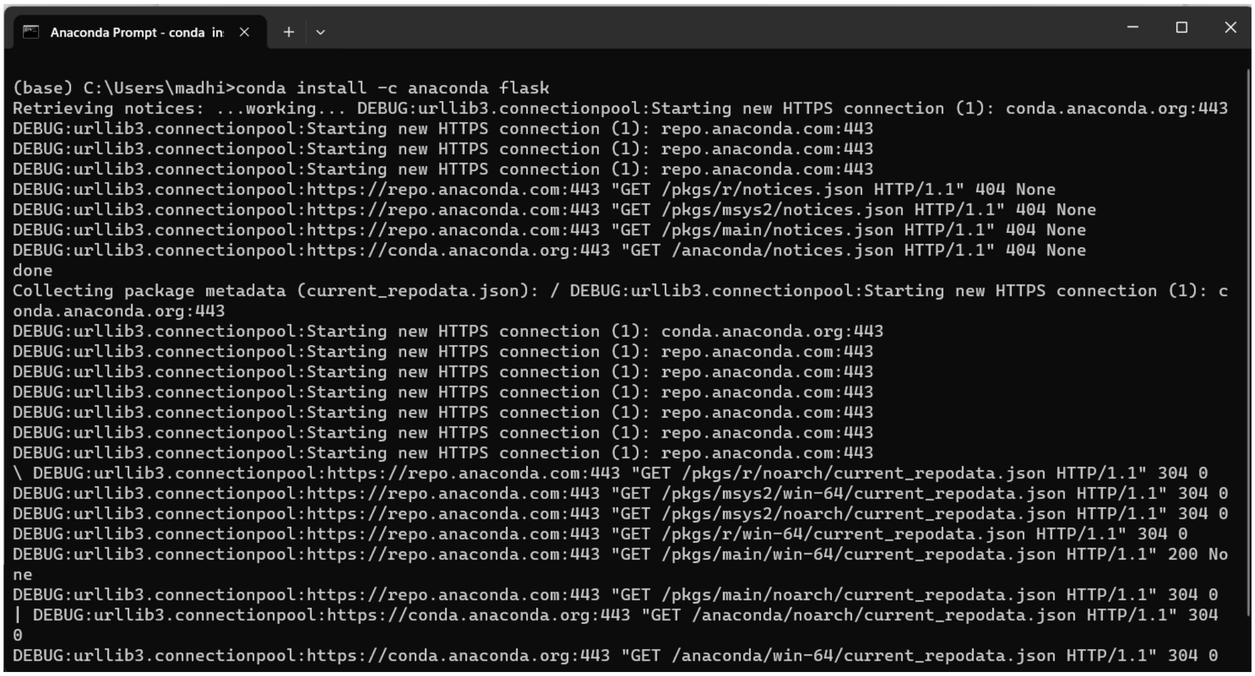
When the application is started as the primary program, it operates in debug mode and displays the forecast on the webpage.

All things considered, this script uses Flask and a trained machine learning model to operate as a functional web application for predicting client turnover.

```python
from flask import Flask, render_template, request
import numpy as np
import pickle
app = Flask(__name__)
best_model_rf = pickle.load(open('best_model_rf.p', 'rb'))
monthly_charges_min, monthly_charges_max = 18.25, 219.0
total_charges_min, total_charges_max = 8684.8, 18.8
gender_map = {'male': 1, 'female': 0}
payment_method_map = {'Bank transfer': 0, 'Credit card': 1, 'Electronic check': 2, 'Mailed check': 3}
yes_no_map = {'yes': 1, 'no': 0}
multiple_lines_map = {'No': 0, 'Yes': 2}
@app.route('/')
def index():
    return render_template('input.html')
@app.route('/predict', methods=['POST'])
def predict():
    gender = gender_map[request.form['gender'].lower()]
    monthly_charges = (float(request.form['MonthlyCharges']) - monthly_charges_min) / (monthly_charges_max - monthly_charges_min)
    payment_method = payment_method_map[request.form['PaymentMethod']]
    paperless_billing = yes_no_map[request.form['PaperlessBilling'].lower()]
    senior_citizen = yes_no_map[request.form['SeniorCitizen'].lower()]
    phone_service = yes_no_map[request.form['PhoneService'].lower()]
    multiple_lines = multiple_lines_map[request.form.get('MultipleLines', 'No')]
    total_charges = (float(request.form['TotalCharges']) - total_charges_min) / (total_charges_max - total_charges_min)
    input_data = [
        gender,
        monthly_charges,
        payment_method,
        paperless_billing,
        senior_citizen,
        phone_service,
        multiple_lines,
        total_charges
    ]
    input_data = np.array([input_data])
    prediction = best_model_rf.predict(input_data)
    if prediction:
        value = 'Yes'
    else:
        value = 'No'
    return render_template('input.html', predicted_value=f'Customer Churn prediction is: {value} ')
if __name__ == '__main__':
    app.run(debug=True)
```

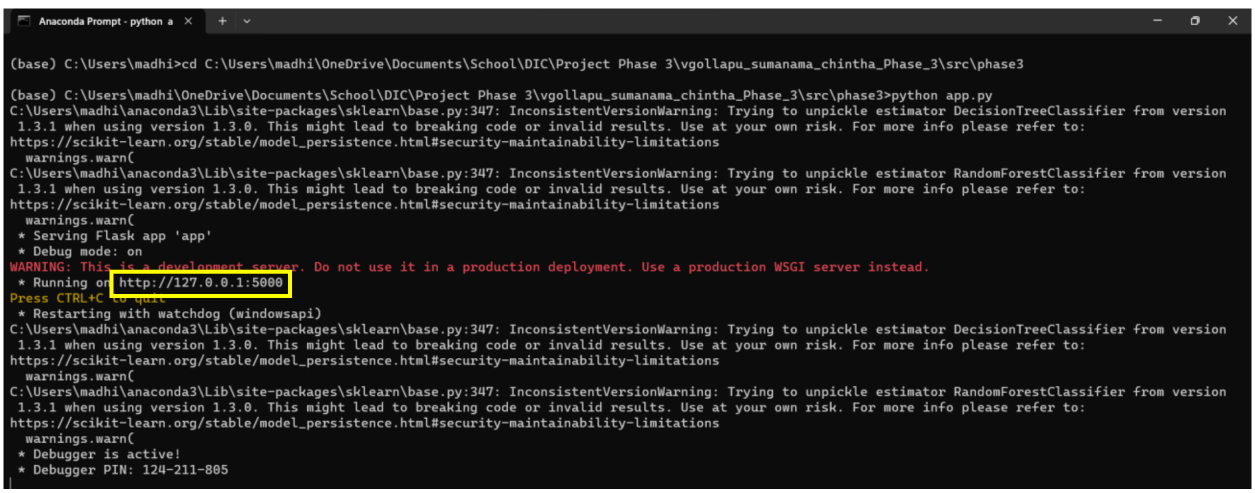
Instructions on how our App works:

Install flask in anaconda prompt



```
Anaconda Prompt - conda in

(base) C:\Users\madhi>conda install -c anaconda flask
Retrieving notices: ...working... DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/notices.json HTTP/1.1" 404 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/notices.json HTTP/1.1" 404 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/notices.json HTTP/1.1" 404 None
DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /anaconda/notices.json HTTP/1.1" 404 None
done
Collecting package metadata (current_repodata.json): / DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): c
onda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): conda.anaconda.org:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
\ DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-64/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/current_repodata.json HTTP/1.1" 200 No
ne
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch/current_repodata.json HTTP/1.1" 304 0
| DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /anaconda/noarch/current_repodata.json HTTP/1.1" 304
0
DEBUG:urllib3.connectionpool:https://conda.anaconda.org:443 "GET /anaconda/win-64/current_repodata.json HTTP/1.1" 304 0
```

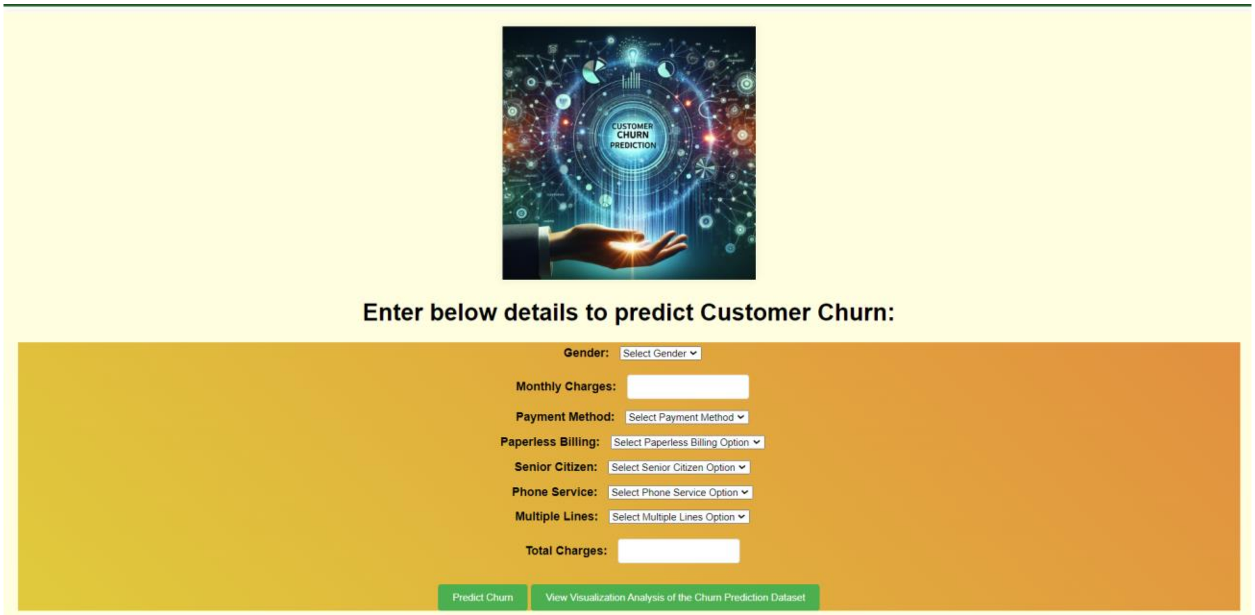
Run the app.py file in anaconda prompt, and copy the url link that we get after running the app.py and open the link in browser to run our predictor app.



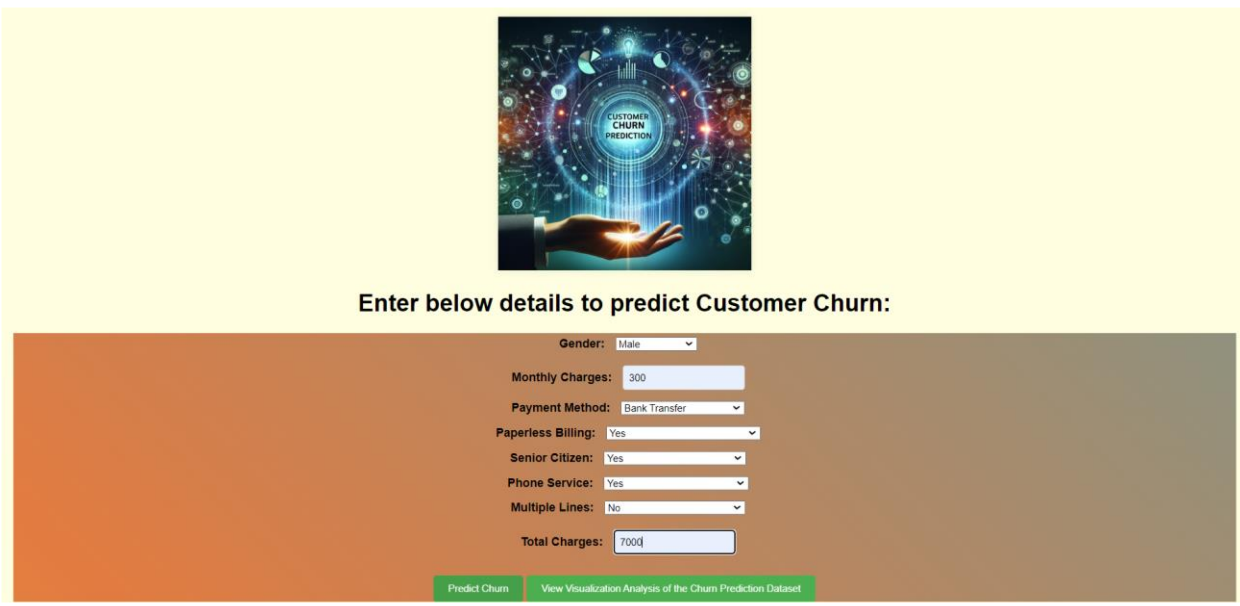
```
Anaconda Prompt - python a

(base) C:\Users\madhi>cd C:\Users\madhi\OneDrive\Documents\School\DIC\Project Phase 3\vgollapu_sumanama_chintha_Phase_3\src\phase3

(base) C:\Users\madhi\OneDrive\Documents\School\DIC\Project Phase 3\vgollapu_sumanama_chintha_Phase_3\src\phase3>python app.py
C:\Users\madhi\anaconda3\Lib\site-packages\sklearn\base.py:347: InconsistentVersionWarning: Trying to unpickle estimator DecisionTreeClassifier from version
 1.3.1 when using version 1.3.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\madhi\anaconda3\Lib\site-packages\sklearn\base.py:347: InconsistentVersionWarning: Trying to unpickle estimator RandomForestClassifier from version
 1.3.1 when using version 1.3.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
C:\Users\madhi\anaconda3\Lib\site-packages\sklearn\base.py:347: InconsistentVersionWarning: Trying to unpickle estimator DecisionTreeClassifier from version
 1.3.1 when using version 1.3.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\madhi\anaconda3\Lib\site-packages\sklearn\base.py:347: InconsistentVersionWarning: Trying to unpickle estimator RandomForestClassifier from version
 1.3.1 when using version 1.3.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 124-211-805
```

http://127.0.0.1:5000

Our Customer Churn Predictor APP

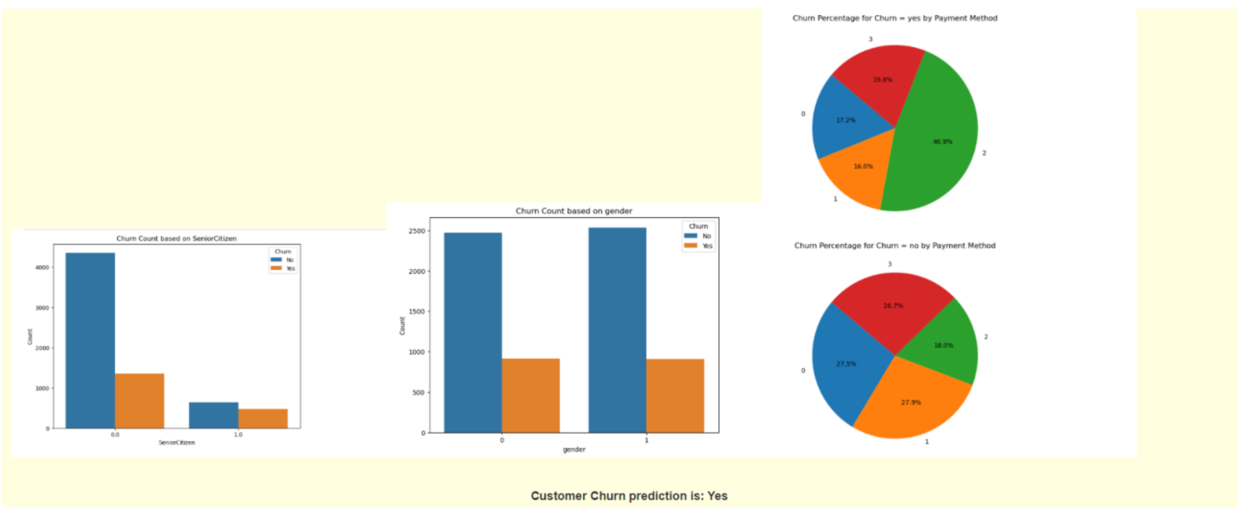


After filling the details in the input fields of customer



After clicking on predict button, we will get the churn prediction for the given details of the customer.

When we click on "View Visualization Analysis of the Churn Prediction Dataset" button, we can view the visualizations of our dataset.



Results

A comprehensive strategy that combines data analytics, customer feedback, and proactive initiatives is essential for telecom companies looking to navigate the challenges posed by churn. This approach not only helps in immediate financial gains but also useful in long-erm customer loyalty.

The way telecom firms handle customer churn, the phenomenon of customers switching providers could be revolutionized by this study. The objective is to offer insightful information by utilizing data and forecasts derived from historical client behavior. Businesses can use this information to anticipate potential employees and take proactive measures to keep them from leaving.

Importantly, telecom providers may make better use of their resources thanks to the model's accuracy and efficiency. They can customize their contacts with clients, addressing certain issues and making them feel more personal. Reducing the number of departing clients is the goal, as it is essential to the success of any telecom company.

It may demonstrate to telecom firms how to apply data analytics in a comparable manner, impacting critical decision-making and enhancing long-term customer connections. The true benefit resides in telecom companies adopting a proactive approach that prevents customer losses rather than one that reacts to them after they happen.