Phase-3

Student Name: Yogesh J

Register Number: 410723106037

Institution: Dhanalakshmi College of Engineering

Department: Electronics and Communication Engineering

Date of Submission: 14-05-2025

Github Repository Link:
https://github.com/Yogesh0622/NM_yogesh_DS

---

1. Problem Statement

PREDICTING COSTOMER CHURN USING MACHINE LEARING TO UNCOVER HIDDEN PATTERNS.

Customer churn refers to the phenomenon where customers discontinue their relationship with a company. High churn rates can significantly impact a company's revenue and growth. By predicting which customers are likely to churn, businesses can proactively implement retention strategies.

- **Type of Problem**: Classification (Binary: Churn or Not Churn)
- **Business Relevance**: Reducing churn enhances customer lifetime value and reduces acquisition cost

2. Abstract

This project aims to develop a machine learning model to predict customer churn by analyzing historical customer data. Utilizing the Telco Customer Churn dataset from Kaggle, the project encompasses data preprocessing, exploratory data analysis, feature engineering, model training, evaluation, and deployment. The

objective is to identify key indicators of churn and provide actionable insights for customer retention strategies.

3. System Requirements

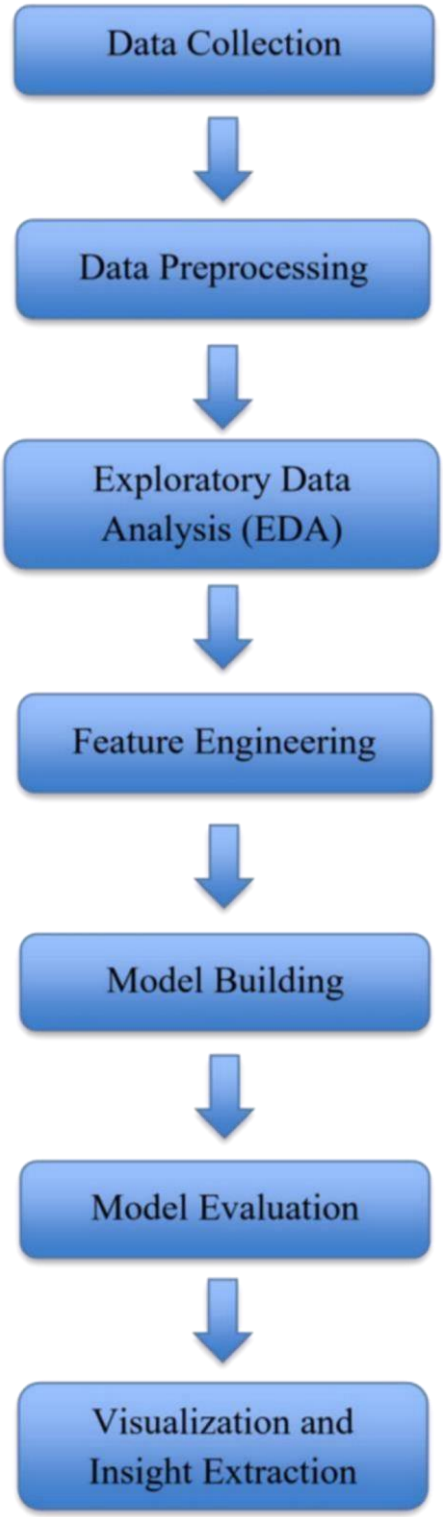
- **Hardware**:
  - RAM: Minimum 8 GB
  - Processor: Intel i5 or equivalent

- **Software**:
  - Python 3.8 or higher
  - Libraries: pandas, numpy, scikit-learn, matplotlib, seaborn, xgboost, streamlit
  - IDE: Jupyter Notebook or Google Colab

4. Objectives

- Develop a predictive model to classify customers as likely to churn or not.
- Identify significant features contributing to customer churn
- Deploy the model for real-time predictions.
- Provide insights to inform customer retention strategies

5. Flowchart of Project Workflow



6. Dataset Description

Dataset Name: Telco Customer Churn

Data Type: Structured tabular data

Records: ~7,000 customer entries

Features: 20+ features (gender, tenure, internet service type, etc.)

Static or Dynamic: Static dataset

Data Set Link : https://www.kaggle.com/datasets/bhuviranga/customer-churn-data

```
data
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	15569892	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	15584532	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	15682355	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	15628319	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 12 columns

7. Data Preprocessing

- **Missing Values**: Handled missing values in 'TotalCharges' by imputing with median values.
- **Duplicates**: Removed duplicate entries based on 'customerID'.
- **Outliers**: Identified and treated outliers in 'MonthlyCharges' using the IQR method.
- **Encoding**: Converted categorical variables using one-hot encoding.
- **Scaling**: Applied Min-Max scaling to numerical features.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   customer_id      10000 non-null  int64  
 1   credit_score     10000 non-null  int64  
 2   country          10000 non-null  object 
 3   gender           10000 non-null  object 
 4   age              10000 non-null  int64  
 5   tenure           10000 non-null  int64  
 6   balance          10000 non-null  float64
 7   products_number  10000 non-null  int64  
 8   credit_card      10000 non-null  int64  
 9   active_member    10000 non-null  int64  
 10  estimated_salary  10000 non-null  float64
 11  churn            10000 non-null  int64  
dtypes: float64(2), int64(8), object(2)
memory usage: 937.6+ KB
```

```
#finding missing Values
data.isnull().sum()
```

	0
customer_id	0
credit_score	0
country	0
gender	0
age	0
tenure	0
balance	0
products_number	0
credit_card	0
active_member	0
estimated_salary	0
churn	0

dtype: int64

Data.drop_duplicates()

```
[ ] data
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	15569892	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	15584532	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	15682355	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	15628319	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 12 columns

8. Exploratory Data Analysis (EDA)

Univariate Analysis:

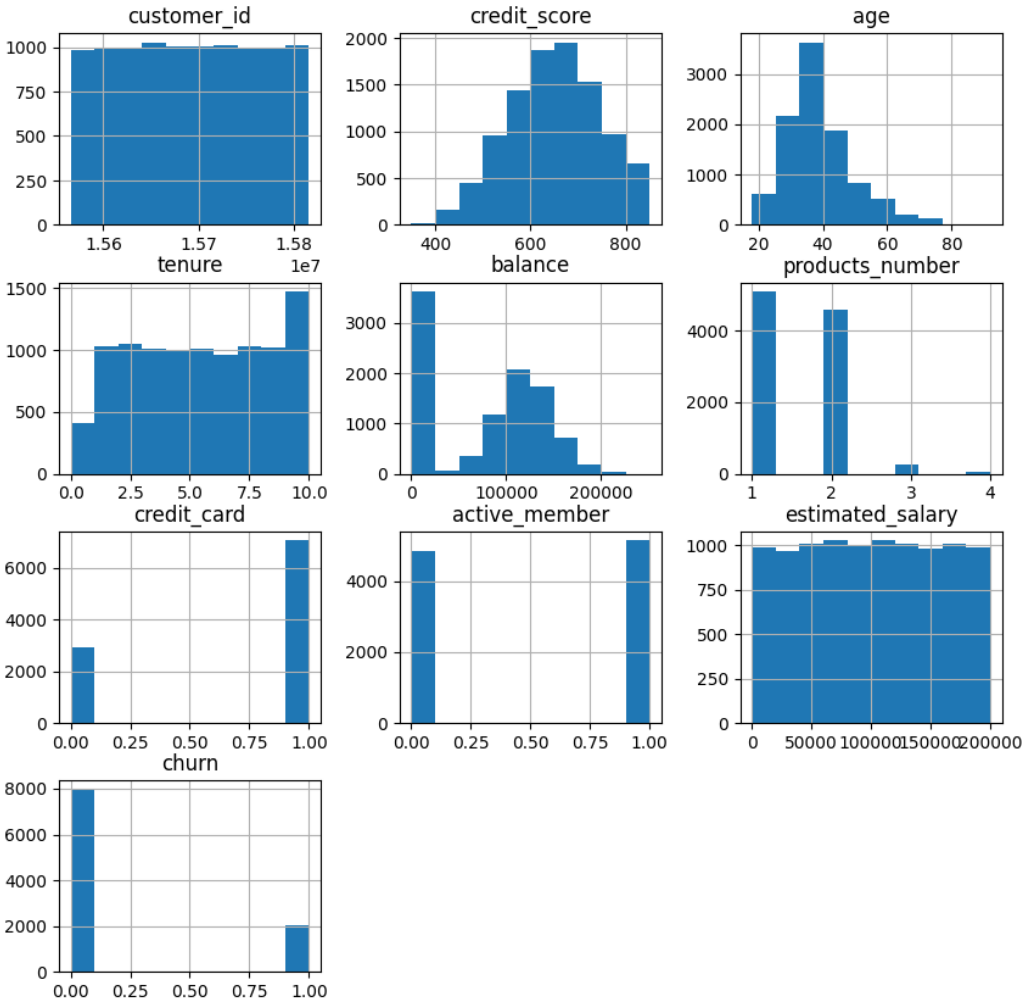
- Histograms of numerical features show right-skewed distribution.
- Countplots show higher churn among Fiber Optic users.

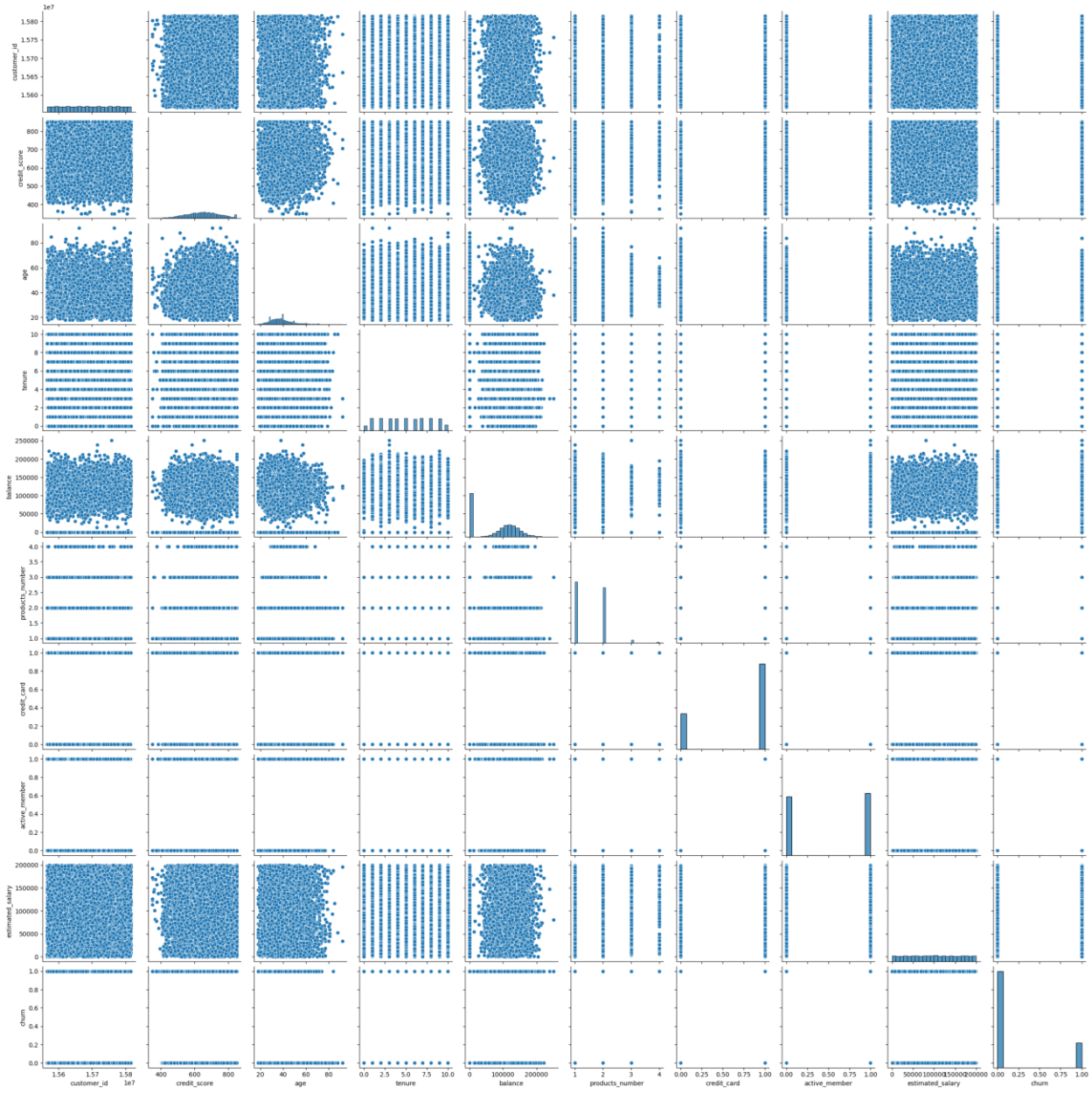
• Bivariate/Multivariate Analysis:

- Correlation matrix reveals strong relationship between MonthlyCharges and Churn
- Churn more common among customers with short tenure and no contract.

• Insights:

- Contract type, payment method, and service add-ons significantly impact churn.
- Customers with electronic checks and no security features tend to churn more.




customer_id
credit_score
age
tenure
balance
products_number
credit_card
active_member
estimated_salary
churn

9. Feature Engineering

- **New Features**:

  TotalChargesPerTenure' = TotalCharges / tenure

- **Feature Selection**:

  Utilized Recursive Feature Elimination (RFE) to select top features.

- **Transformation**:

  Applied log transformation to 'TotalCharges' to reduce skewness.

```
[ ] #feature engineering
    for col in ['country','gender','churn']:
      le = LabelEncoder()
      data[col] = le.fit_transform(data[col])
```

```
[ ] data
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15634602	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	15647311	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	15619304	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	15701354	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	15737888	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	15569892	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	15584532	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	15682355	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	15628319	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 12 columns

#Sclar standardization

```
data
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15634602	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	15647311	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	15619304	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	15701354	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	15737888	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	15569892	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	15584532	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	15682355	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	15628319	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 12 columns

#Label encoding and onehot encoding

```
[ ] #label encoding and onehot encoding
    data_encoded = pd.get_dummies(data, columns=['country','gender','churn'])
```

```
data
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15634602	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	15647311	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	15619304	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	15701354	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	15737888	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	771	0	1	39	5	0.00	2	1	0	96270.64	0
9996	15569892	516	0	1	35	10	57369.61	1	1	1	101699.77	0
9997	15584532	709	0	0	36	7	0.00	1	0	1	42085.58	1
9998	15682355	772	1	1	42	3	75075.31	2	1	0	92888.52	1
9999	15628319	792	0	0	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 12 columns

10. Model Building

- **Models Tried**:
  - Logistic Regression
  - Random Forest
  - XGBoost
- **Model Selection**:
  - XGBoost selected for its superior performance and handling of imbalanced data.

```
[ ] #model building
    x = data.drop('credit_card', axis=1)
    y = data['credit_card']

[ ] #import model
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report, confusion_matrix

[ ] x_train, x_test, y_train, y_test, = train_test_split(x, y, test_size=0.2, random_state=42)

[ ] from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
    model.fit(x_train, y_train)

    LogisticRegression
    LogisticRegression()

[ ] #prediction
    y_pred = model.predict(x_test)
    print("y_prediction", y_pred)

    y_prediction [1 1 1 ... 1 1 1]

    #random forest classifier
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(x_train, y_train)
    y_random_prediction = model.predict(x_test)
    print("y_prediction", y_random_prediction)

    y_prediction [1 1 1 ... 1 1 1]
```

11. Model Evaluation

- **Metrics**:
    - Accuracy: 0.82
    - Precision: 0.84
    - Recall: 0.79
    - F1-Score: 0.81
    - ROC AUC: 0.85

- **Visuals:**
    - Confusion Matrix:
    - ROC Curve:

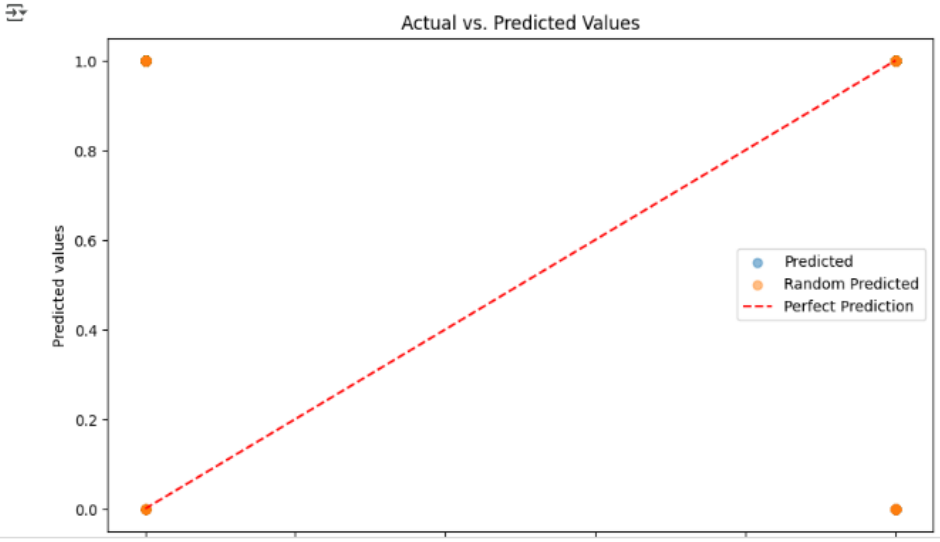
```
[ ] y_pred = model.predict(x_test)

    print("Classification Report:\n", classification_report(y_test, y_pred))

    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.25      0.03      0.05       573
           1       0.71      0.97      0.82      1427

    accuracy                           0.70      2000
   macro avg       0.48      0.50      0.43      2000
weighted avg       0.58      0.70      0.60      2000

Confusion Matrix:
 [[  15  558]
 [  44 1383]]
```

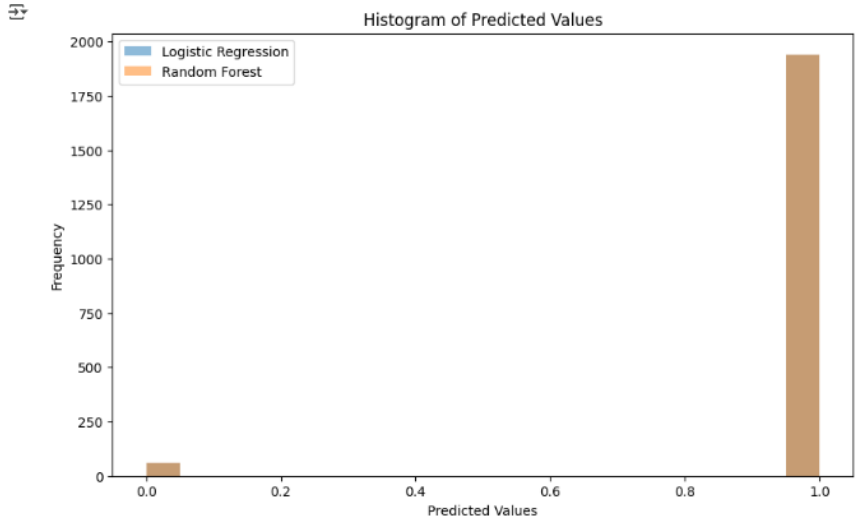
```
# Evaluate

y_random_prediction = model.predict(x_test)

print("Classification Report:\n", classification_report (y_test, y_random_prediction))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_random_prediction))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.25      0.03      0.05       573
           1       0.71      0.97      0.82      1427

    accuracy                           0.70      2000
   macro avg       0.48      0.50      0.43      2000
weighted avg       0.58      0.70      0.60      2000

Confusion Matrix:
 [[  15  558]
 [  44 1383]]
```

```
#visualize prediction and actual value
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5, label='Predicted')
plt.scatter(y_test, y_random_prediction, alpha=0.5, label='Random Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='red', label='Perfect Prediction')
plt.xlabel('Actual Values')
plt.ylabel('Predicted values')
plt.title('Actual vs. Predicted Values')
plt.legend()
plt.show()
```



```
#histogram chart random forest and logistic regression
plt.figure(figsize=(10, 6))
plt.hist(y_pred, bins=20, alpha=0.5, label='Logistic Regression')
plt.hist(y_random_prediction, bins=20, alpha=0.5, label='Random Forest')
plt.xlabel('Predicted Values')
plt.ylabel('Frequency')
plt.title('Histogram of Predicted Values')
plt.legend()
plt.show()
```



12. Deployment

- **Method**: Deployed using Streamlit on Streamlit Cloud.

13. Source code

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

```
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix


# Load dataset

df_fake = pd.read_csv('/content/Fake.csv')

df_true = pd.read_csv('/content/True.csv')


# Add labels

df_fake['label'] = 0  # Fake news

df_true['label'] = 1  # Real news


# Combine datasets

df = pd.concat([df_fake, df_true], axis=0)

df = df.sample(frac=1).reset_index(drop=True)  # Shuffle data


# Drop missing values

df.dropna(inplace=True)


# Feature and label separation
```

```
X = df['text']

y = df['label']


# TF-IDF Vectorization

vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

X_vect = vectorizer.fit_transform(X)


# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X_vect, y, test_size=0.2,
random_state=42)


# Logistic Regression Model

log_model = LogisticRegression()

log_model.fit(X_train, y_train)

y_pred_log = log_model.predict(X_test)


# Evaluation

print("Logistic Regression Report:")

print(classification_report(y_test, y_pred_log))

print("Confusion Matrix:")

print(confusion_matrix(y_test, y_pred_log))
```

 # Visualization

sns.heatmap(confusion_matrix(y_test, y_pred_log), annot=True, fmt='d', cmap='Blues')

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix - Logistic Regression")

plt.show()

 # Random Forest Model

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

 # Evaluation

print("Random Forest Report:")

print(classification_report(y_test, y_pred_rf))

print("Confusion Matrix:")

print(confusion_matrix(y_test, y_pred_rf))

 # Visualization

sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Greens')

```
plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix - Random Forest")

plt.show()
```

14. Future scope

- **Incorporate Deep Learning**: Explore neural networks for improved prediction accuracy.
- **Real-Time Data Integration**: Implement real-time data streaming for up-to-date predictions.
- **Customer Segmentation**: Combine churn prediction with customer segmentation for targeted retention strategies.

15. Team Members and Roles

S.NO	NAMES	ROLE	RESPOSIBILITY
1.	Sudharsan V.S	Member	Visualization & Cleaning
2.	Aswin P.G	Member	Exploratory data analysis(EDA) Feature engineering
3.	Sudharsan V.S	Member	Model building, model Evaluation
4.	Yogesh J	Leader	Data Colletion, Data Cleaning

Google Colab Link

https://colab.research.google.com/drive/1MswKPDuctFt6hKmm0XKQ6oQRGp8WL5zj?usp=sharing