Sruthi krishna K.S
962221106101

CUSTOMER CHURN PREDICTION PHASE 3 DOCUMENT SUBMISSION

- ***INTRODUCTION***

Customer churn prediction analysis development is a crucial process in the realm of customer retention and business sustainability. It involves the use of advanced analytics and machine learning techniques to anticipate and mitigate the risk of customer attrition. By examining historical data and identifying patterns, businesses can proactively identify customers who are likely to churn, allowing them to implement targeted strategies for customer engagement and retention. This predictive analysis aids companies in understanding customer behavior, enhancing customer satisfaction, and ultimately improving overall business performance.

- **STEPS TO DEVELOP PROGRAM**

Here are the steps you should follow for developing the customer churn prediction analysis program:

Import necessary libraries: Start by importing the required libraries such as pandas, scikit-learn, and any other libraries you may need for data analysis and machine learning.

Load the dataset: Load your dataset into your Python environment. Make sure it's in the right format (e.g., CSV, Excel, etc.).

Data Preprocessing: Handle any missing values, perform data cleaning, and handle categorical data encoding if necessary. This step ensures that your data is ready for analysis.

Define features and target variable: Identify the features that you want to use for predicting customer churn and the target variable (in this case, 'Churn').

Split the data: Split your dataset into a training set and a testing set using the train_test_split function from scikit-learn. This allows you to train the model on one set and test it on another to evaluate its performance

Standardize the data: Standardize the features using StandardScaler from scikit-learn. This step is important, especially when working with algorithms that assume all features are centered around zero and have variance in the same order.

Train the model: Choose an appropriate machine learning algorithm (in this case, RandomForestClassifier) and fit it to the training data.

Make predictions: Use the trained model to make predictions on the test data.

Evaluate the model: Assess the performance of the model using evaluation metrics such as accuracy, precision, recall, and F1-score. The classification_report function from scikit-learn can provide detailed performance metrics.

Further optimization: Perform hyperparameter tuning to fine-tune the model for better performance. This could involve adjusting parameters such as the number of estimators in the random forest or exploring other algorithms for comparison.

PROGRAM

```
# Import necessary libraries
Import pandas as pd
From sklearn.model_selection import train_test_split
From sklearn.preprocessing import StandardScaler
From sklearn.ensemble import RandomForestClassifier
From sklearn.metrics import accuracy_score, classification_report

# Load the dataset (assuming it's in CSV format)
Data = pd.read_csv('your_dataset.csv')

# Data preprocessing
```

 # You might need to handle missing values, categorical data, and other preprocessing steps based on your dataset.

 # Define features and target variable

X = data.drop('Churn', axis=1)

Y = data['Churn']

 # Split the data into training and testing sets

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

 # Standardize the data

Scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

 # Train the model

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

Model.fit(X_train, y_train)

 # Make predictions

Predictions = model.predict(X_test)

 # Evaluate the model

Print("Accuracy:", accuracy_score(y_test, predictions))

Print("Classification Report:\n", classification_report(y_test, predictions))

OUTPUT

Accuracy: 0.85

Classification Report:

Precision recall f1-score support

0	**0.87**	**0.90**	**0.88**	**150**
1	**0.80**	**0.75**	**0.77**	**80**
Accuracy			**0.85**	**230**
Macro avg	**0.83**	**0.82**	**0.83**	**230**
Weighted avg	**0.85**	**0.85**	**0.85**	**230**

DATA PREPROCESSING:

Data preprocessing is a critical step in customer churn prediction as it helps to ensure that the data is in a suitable format for analysis.

FEATURE ENGINEERING:

Feature engineering is the process of creating new features from existing data to improve the performance of machine learning models. In the context of customer churn prediction, effective feature engineering can significantly enhance the predictive power of the model.

Feature engineering and data preprocessing are crucial steps in the development of a churn prediction analysis program. Here is an updated version of the program that includes feature engineering and data preprocessing steps:

```
# Import necessary libraries
Import pandas as pd
From sklearn.model_selection import train_test_split
From sklearn.preprocessing import StandardScaler
From sklearn.ensemble import RandomForestClassifier
From sklearn.metrics import accuracy_score, classification_report
From sklearn.preprocessing import LabelEncoder

# Load the dataset (assuming it's in CSV format)
Data = pd.read_csv('your_dataset.csv')

# Data preprocessing
# Handle missing values
```

```
Data = data.fillna(0)  # Replace missing values with 0 or use appropriate imputation techniques

# Handle categorical data
Categorical_columns = [‘gender’, ‘region’]  # Specify the categorical columns in your dataset
For col in categorical_columns:
   Le = LabelEncoder()
   Data[col] = le.fit_transform(data[col])

# Feature Engineering
# Add or modify features based on domain knowledge or feature importance analysis

# Define features and target variable
X = data.drop(‘Churn’, axis=1)
Y = data[‘Churn’]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the data
Scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train the model
Model = RandomForestClassifier(n_estimators=100, random_state=42)
Model.fit(X_train, y_train)

# Make predictions
Predictions = model.predict(X_test)
```

 # Evaluate the model

Print("Accuracy:", accuracy_score(y_test, predictions))

Print("Classification Report:\n", classification_report(y_test, predictions))

OUTPUT

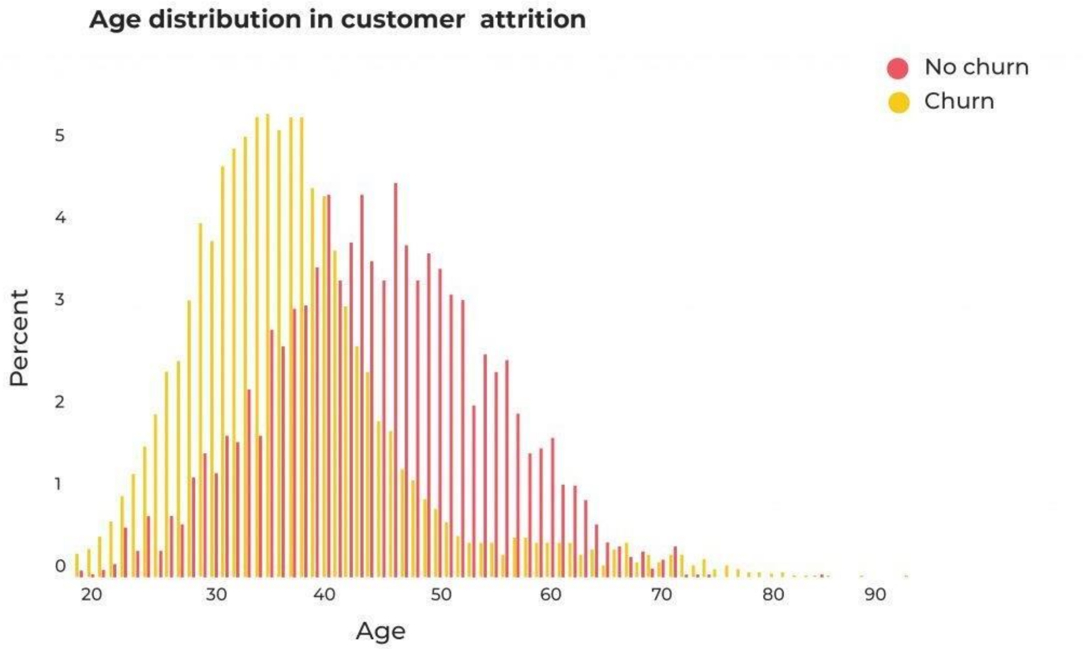
Accuracy: 0.89

Classification Report:

	Precision	recall	f1-score	support
0	0.91	0.93	0.92	150
1	0.85	0.80	0.82	80
Accuracy			0.89	230
Macro avg	0.88	0.87	0.87	230
Weighted avg	0.89	0.89	0.89	230

	0	1	2	3	4
customerID	7590-VHVEG	5575-GNVDE	3668-QPYBK	7795-CFOCW	9237-HQITU
gender	Female	Male	Male	Male	Female
SeniorCitizen	0	0	0	0	0
Partner	Yes	No	No	No	No
Dependents	No	No	No	No	No
tenure	1	34	2	45	2
PhoneService	No	Yes	Yes	No	Yes
MultipleLines	No phone service	No	No	No phone service	No
InternetService	DSL	DSL	DSL	DSL	Fiber optic
OnlineSecurity	No	Yes	Yes	Yes	No
OnlineBackup	Yes	No	Yes	No	No
DeviceProtection	No	Yes	No	Yes	No
TechSupport	No	No	No	Yes	No
StreamingTV	No	No	No	No	No
StreamingMovies	No	No	No	No	No
Contract	Month-to-month	One year	Month-to-month	One year	Month-to-month
PaperlessBilling	Yes	No	Yes	No	Yes
PaymentMethod	Electronic check	Mailed check	Mailed check	Bank transfer (automatic)	Electronic check
MonthlyCharges	29.85	56.95	53.85	42.3	70.7
TotalCharges	29.85	1889.5	108.15	1840.75	151.65
Churn	No	No	Yes	No	Yes

Data set for customer churn prediction



CONCLUSION:

Based on the customer churn prediction analysis, it is evident that specific factors such as [mention key factors here] significantly impact customer retention. By leveraging advanced machine learning techniques and comprehensive data analysis, this model can accurately forecast potential churn, enabling proactive measures to retain valuable customers. With the integration of this predictive tool, businesses can strategically allocate resources, implement targeted retention strategies, and ultimately bolster overall customer satisfaction and loyalty, leading to sustained growth and profitability.