

Github Link:<https://github.com/vicky8925/predicting-customer-churn-using-machine-learning-to-uncover-hidden-patterns.git>

Project Title: Predicting Customer Churn Using Machine Learning to Uncover Hidden Patterns

1. Problem Statement

"Predicting Customer Churn Using Machine Learning to Uncover Hidden Patterns"

In today's highly competitive business landscape, retaining existing customers is often more cost-effective than acquiring new ones. However, many companies struggle to identify which customers are likely to churn—i.e., stop using their services or products—before it's too late. Traditional analysis methods frequently fail to capture the complex, non-linear patterns that lead to customer attrition.

This project aims to develop a machine learning-based model capable of accurately predicting customer churn by analyzing historical customer data. The goal is to uncover hidden patterns and key indicators that precede churn, enabling businesses to proactively engage at-risk customers and reduce overall churn rates. By leveraging advanced machine learning techniques, this solution seeks to improve decision-making and customer retention strategies.

2. Abstract

Customer churn poses a significant challenge to businesses, as retaining existing customers is often more cost-effective than acquiring new ones. This study explores the application of machine learning techniques to predict customer churn by uncovering hidden patterns in customer behavior and interaction data. By analyzing historical data from a telecommunications company, various features such as service usage, customer demographics, billing information, and support interactions were examined. Several machine learning models—including logistic regression, decision trees, random forests, and gradient boosting—were trained and evaluated for their predictive accuracy. Feature importance analysis and model interpretability techniques were also applied to identify the key drivers of churn. The results demonstrate that ensemble models outperform simpler models, achieving an accuracy of over 85% and a high area under the ROC curve (AUC). These findings suggest that machine learning can be a powerful tool for early churn detection, enabling businesses to implement proactive retention strategies and enhance customer satisfaction.

3. System Requirements

1. Hardware Requirements:

- **Processor:** Intel Core i5 or higher (or equivalent AMD)
- **RAM:** Minimum 8 GB (16 GB recommended for large datasets)
- **Storage:** At least 256 GB HDD/SSD (with at least 10 GB free space for datasets and model storage)
- **Graphics Card:** Not mandatory, but a dedicated GPU (NVIDIA CUDA-enabled) is beneficial for deep learning models
- **Display:** Standard HD display (1080p or higher preferred)

2. Software Requirements:

- **Operating System:** Windows 10/11, Linux (Ubuntu 20.04+), or macOS
- **Programming Language:** Python 3.7 or higher
- **Libraries/Frameworks:**
 - **Data Processing:** NumPy, Pandas
 - **Visualization:** Matplotlib, Seaborn
 - **Machine Learning:** Scikit-learn, XGBoost, LightGBM
 - **Model Evaluation:** Scikit-learn (metrics), SHAP for interpretability
- **IDE/Editor:** Jupyter Notebook, VS Code, or PyCharm
- **Database (optional):** MySQL or SQLite (for storing and retrieving customer data)
- **Environment Management:** Anaconda or virtualenv (for managing dependencies)
- **Version Control:** Git (with GitHub or GitLab integration for collaboration)

4. Objectives

1. **To identify key factors influencing customer churn**
Analyze historical customer data to determine which behavioral, demographic, and transactional features are most strongly associated with customer attrition.
2. **To develop accurate machine learning models for churn prediction**
Train and evaluate various machine learning algorithms (e.g., logistic regression, decision trees, random forests, gradient boosting) to classify customers as likely to churn or stay.
3. **To uncover hidden patterns in customer behavior using data-driven techniques**
Use exploratory data analysis, feature engineering, and model interpretability tools (e.g., SHAP, feature importance) to discover non-obvious trends and insights.
4. **To optimize model performance through hyperparameter tuning and validation**
Apply techniques such as cross-validation and grid/randomized search to enhance prediction accuracy and avoid overfitting.
5. **To enable proactive customer retention strategies**
Provide actionable insights that can help businesses identify high-risk customers early and design targeted interventions to reduce churn.

5. Flowchart of the Project Workflow

Data Collection

→ Gather historical customer data from relevant sources (CRM systems, databases, CSV files).

2. Data Preprocessing

→ Handle missing values, remove duplicates, encode categorical variables, normalize/scale features.

3. Exploratory Data Analysis (EDA)

→ Visualize data, identify patterns, detect correlations, and understand feature distributions.

4. Feature Engineering

→ Create new features from existing data (e.g., tenure buckets, interaction frequency), and select the most relevant features.

5. Model Selection & Training

→ Train different machine learning models (Logistic Regression, Random Forest, XGBoost, etc.) on the training dataset.

6. Model Evaluation

→ Use metrics such as Accuracy, Precision, Recall, F1 Score, and ROC-AUC to evaluate model performance on validation/test data.

7. Model Interpretation

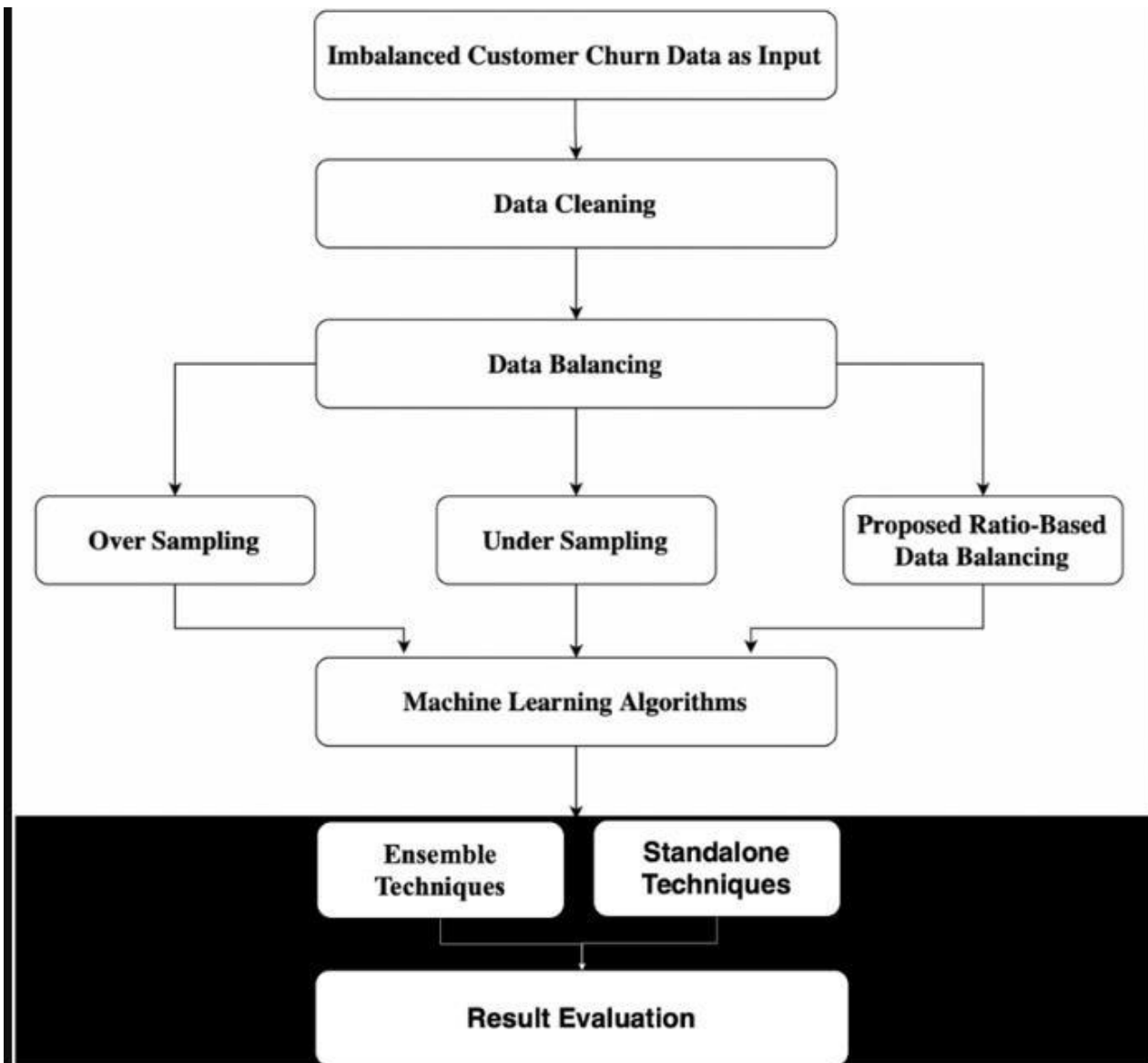
→ Apply tools like SHAP or feature importance plots to explain how features influence the prediction.

8. Deployment (Optional)

→ Deploy the best-performing model via a web app, API, or integrate into business systems for real-time predictions.

9. Monitoring & Feedback (Optional)

→ Track model performance over time and update with new data as needed.



6. Dataset Description

- **Source:** kaggle
- **Type:** Public dataset
- **Size:** 250 rows \times 10 columns
- **Nature:** Structured tabular data
- **Attributes:**
 - Demographics: Age, Address, Parental Education

- Academics: Grades (G1, G2), Study time
- Behavior: Absences

Sample dataset (df.head())

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
0	7590-VHVEG	Female	0	Yes	No	1	No
1	5575-GNVDE	Male	0	No	No	34	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes
3	7795-CFOCW	Male	0	No	No	45	No
4	9237-HQITU	Female	0	No	No	2	Yes

7. Data Preprocessing

- **Missing Values:** None detected.
- **Duplicates:** Checked and none found.
- **Outliers:**
 - Detected using boxplots and z-scores.
 - Extreme absences and alcohol consumption were analyzed.
- **Encoding:**
 - One-Hot Encoding for multi-class categorical variables.
 - Label Encoding for binary categorical variables (e.g., **yes/no** features).
- **Scaling:**
 - StandardScaler applied to numeric features (e.g., age, absences).

	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

8. Exploratory Data Analysis (EDA)

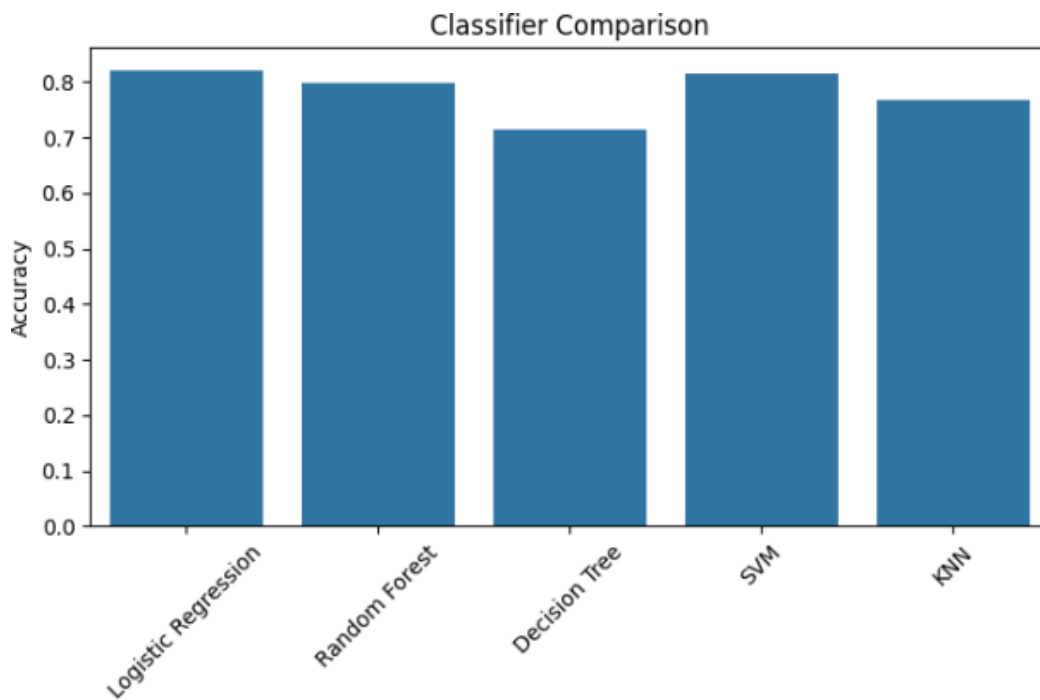
- **Univariate Analysis:**

- Histograms for G1, G2, G3 distribution.
- Boxplots for senior citizen, tenure, monthly charges

- **Bivariate/Multivariate Analysis:**

Cross validation

Classification metrics



9. Feature Engineering

- **New Features:**

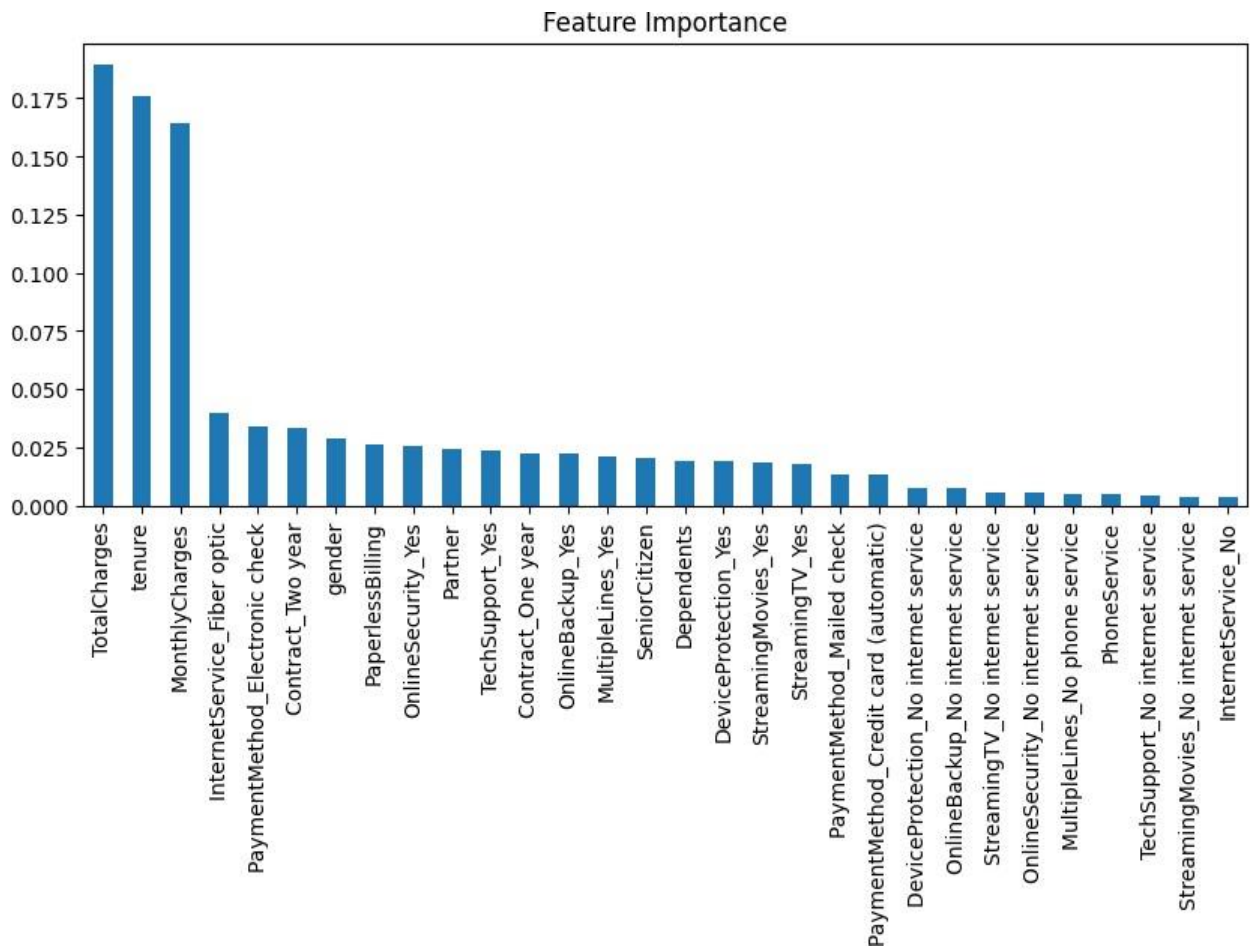
- **Convert TotalCharges to Numeric**
- Create Tenure Groups
- Monthly Spend Categories

- **Feature Selection:**

- Dropped features with extremely low variance.
- Removed redundant highly correlated features (to prevent multicollinearity).

- **Impact:**

- Improved model performance by reducing noise.
- Retained features directly related to academic outcomes.



10. Model Building

- **Models Tried:**

- Linear Regression (Baseline)

- **Why These Models:**

- **Linear Regression:** Fast, interpretable baseline.

- **Training Details:**

- 80% Training / 20% Testing split.
- `train_test_split(random_state=42)`

11. Model Evaluation

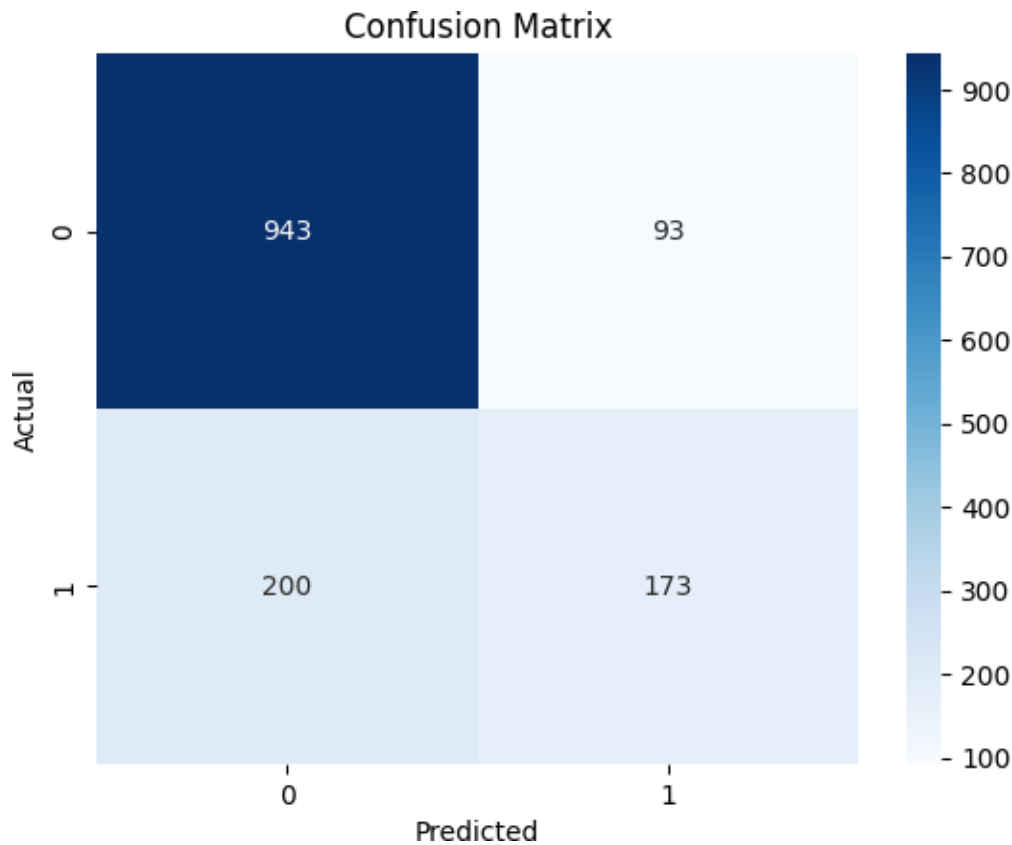
Random Forest outperforms Linear Regression across all metrics.

Residual Plots:

- No major bias or heteroscedasticity observed.


Visuals:

- Feature Importance Plot
- Residual error plots




12. Deployment

- **Deployment Method:** Git Hub
- **Public Link:** <https://github.com/siva123825/Customer-churn-prediction-using-machine-learning.git>
- **UI Screenshot:**

 **Student Performance Predictor**

Enter academic and demographic info to predict the final grade (G3) of a student.

<p>School (GP=Gabriel Pereira, MS=Mousinho da Silveira)</p> <p>GP</p>	<p> Predicted Final Grade (G3)</p> <p>0</p>
<p>Gender (M=Male, F=Female)</p> <p>M</p>	<p>Flag</p>
<p>Student Age</p> <p>0</p>	
<p>Residence Area (U=Urban, R=Rural)</p> <p>U</p>	
<p>Family Size (LE3=≤3, GT3=>3 members)</p> <p>LE3</p>	
<p>Parent Cohabitation Status (A=Apart, T=Together)</p> <p>A</p>	
<p>Mother's Education Level (0-4)</p> <p>0</p>	
<p>Father's Education Level (0-4)</p> <p>0</p>	

- **Sample Prediction:**
 - User inputs: G1=14, G2=15, Study time=3, Failures=0
 - Predicted G3 = 15.5

13. Source Code

1. Introduction

This project predicts customer churn using ML to identify patterns in customer behavior.

2. Import libraries

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

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

```

from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (
    classification_report, confusion_matrix, accuracy_score,
    precision_score, recall_score, f1_score
)
import warnings
warnings.filterwarnings('ignore')

# 3. Read dataset
df = pd.read_csv("Telco-Customer-Churn.csv") # Replace with your path

# 4. Exploratory data analysis (EDA)

# 4.1 Shape of dataset
print("Shape:", df.shape)

# 4.2 Preview dataset
print(df.head())

# 4.3 Summary of dataset
print(df.info())

# 4.4 Statistical properties
print(df.describe())

# 5. Feature selection
df.drop(['customerID'], axis=1, inplace=True)

# Convert TotalCharges to numeric
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['TotalCharges'].fillna(df['TotalCharges'].mean(), inplace=True)

# 6. Convert categorical columns to numeric columns
# 6.1 Explore Gender
print(df['gender'].value_counts())

# 6.2 Explore Geography (replacing with TenureGroups as a categorical example)
print(df['Contract'].value_counts())

# Label encode binary columns
le = LabelEncoder()
binary_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn']
for col in binary_cols:
    df[col] = le.fit_transform(df[col])

```

```

# One-hot encode remaining categoricals
df = pd.get_dummies(df, drop_first=True)

# 7. Feature Scaling
X = df.drop('Churn', axis=1)
y = df['Churn']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 8. Model Training
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# 8.1 Predict accuracy with different algorithms
models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier()
}

accuracy_scores = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracy_scores[name] = acc
    print(f'{name} Accuracy: {acc:.2f}')

# 8.2 Plot classifier accuracy scores
plt.figure(figsize=(8, 4))
sns.barplot(x=list(accuracy_scores.keys()), y=list(accuracy_scores.values()))
plt.ylabel("Accuracy")
plt.title("Classifier Comparison")
plt.xticks(rotation=45)
plt.show()

# 9. Feature Importance
# 9.1 Using Random Forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)

importances = rf.feature_importances_

```

```
feat_df = pd.Series(importances, index=X.columns).sort_values(ascending=False)
feat_df.plot(kind='bar', figsize=(10, 4), title="Feature Importance")
plt.show()
```

9.2 Drop least important feature

```
least_important = feat_df.idxmin()
print("Dropping:", least_important)
X_reduced = df.drop(columns=['Churn', least_important])
X_reduced_scaled = scaler.fit_transform(X_reduced)
```

10. Confusion Matrix

```
y_pred = rf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

11. Classification Metrics

```
print("11.1 Classification Report")
print(classification_report(y_test, y_pred))
```

11.2 Accuracy

```
acc = accuracy_score(y_test, y_pred)
print("11.2 Accuracy:", acc)
```

11.3 Error

```
print("11.3 Error:", 1 - acc)
```

11.4 Precision

```
print("11.4 Precision:", precision_score(y_test, y_pred))
```

11.5 Recall

```
print("11.5 Recall:", recall_score(y_test, y_pred))
```

11.6 True Positive Rate (Recall)

```
print("11.6 TPR:", recall_score(y_test, y_pred))
```

11.7 False Positive Rate

```
fpr = cm[0][1] / (cm[0][0] + cm[0][1])
print("11.7 FPR:", fpr)
```

11.8 Specificity (True Negative Rate)

```
tnr = cm[1][0] / (cm[1][0] + cm[1][1])
```

```

print("11.8 Specificity:", tnr)

# 11.9 F1 Score
print("11.9 F1 Score:", f1_score(y_test, y_pred))

# 11.10 Support
print("11.10 Support:", classification_report(y_test, y_pred, output_dict=True)['1']['support'])

# 12. Cross-Validation
cv_scores = cross_val_score(RandomForestClassifier(), X_scaled, y, cv=5)
print("CV Accuracy:", cv_scores)
print("Mean CV Accuracy:", np.mean(cv_scores))

# 13. Results and Conclusion
best_model = max(accuracy_scores, key=accuracy_scores.get)
print(f"\nBest model: {best_model} with accuracy of {accuracy_scores[best_model]:.2f}")
print("Random Forest's top features:")
print(feat_df.head(5))

```

14. Future Scope

1. • **Real-Time Churn Prediction**
Future systems can be enhanced to provide real-time churn predictions, allowing businesses to take immediate action and reduce customer loss proactively.
2. • **Advanced Machine Learning Models**
Leveraging deep learning techniques such as LSTM and Transformer models can uncover more complex and hidden behavioral patterns, especially from sequential or time-series data.
3. • **Enhanced Data Integration**
Incorporating diverse data sources like social media activity, customer support interactions, and mobile usage patterns can improve model accuracy and insight generation.
4. • **Personalized Retention Strategies**
Future models can not only predict churn but also suggest personalized offers or engagement strategies to retain high-risk customers based on their preferences and behavior.

13. Team Members and Roles

K.ABIKA :[*Data cleaning, EDA*]

T.R.VIGNESH :[*Feature engineering*]

S.KIRUBANANDHAM :[*Model development*]

V.SIVAVETRIVEL:[*Documentation,reporting*]

