DATA MINING

SPRING 2025

COURSE PROJECT

<Bank Customer Churn Prediction Using Decision Tree Algorithm>

Night Ed. **Group No : 5**

Emrecan AKGÜL- 210316035

Novruz NAJAFLI- 200316077

Nejat KARAPINAR- 220316080

20 May 2025

Contents

I.Introduction

This report presents a comprehensive machine learning workflow to predict customer churn in a bank setting. Churn prediction is vital for customer retention and profitability. This study applies data preprocessing, SMOTE balancing, Decision Tree modeling, hyperparameter optimization, and evaluation using performance metrics and real-time predictions through both batch and interactive methods.

About Decision Tree Algorithm

A **Decision Tree** is a supervised learning algorithm used for both classification and regression tasks. It operates by recursively splitting the dataset into subsets based on feature values, creating a tree-like structure of decision rules. Each internal node represents a condition on a feature, each branch corresponds to the outcome of that condition, and each leaf node represents a final class label or value.

Advantages of Decision Trees

- **Interpretability** easy to understand and interpret.
- **No Feature Scaling Required:** Unlike distance-based models, they do not require feature normalization.
- **Handles Non-Linear Relationships:** Able to capture complex, non-linear patterns in data.
- **Works with Both Numerical and Categorical Data:** They naturally handle mixed-type variables.
- **Minimal Data Preparation:** No need for dummy variables or imputation assumptions.

Disadvantages of Decision Trees

- **Overfitting Risk:** Trees can become overly complex and fit noise in the training data if not pruned.
- **Instability:** Small changes in data can lead to different tree structures.
- **Bias Toward Features with More Levels:** Trees may favor features with more splits, which can distort learning.
- **Limited Generalization:** Single trees generally do not perform as well as ensemble methods (e.g., Random Forests).

Why Decision Tree Was Chosen for This Project

In the context of customer churn prediction, **Decision Trees** were selected due to their high interpretability, which is crucial for understanding customer behavior and justifying

predictions to business stakeholders. They offer a clear path from input features (like credit score, geography, or balance) to the final prediction (churn or not), which aids in transparency.

Moreover, Decision Trees work well even when the data contains both categorical and numerical attributes, aligning perfectly with the mixed-format nature of the bank customer dataset. While more complex models (like ensemble methods) could offer slight performance improvements, the Decision Tree algorithm balances accuracy with simplicity and explainability, making it an ideal starting point for deployment or integration with business decision systems.

2.Dataset Description and Preprocessing

The dataset consists of 165,034 customer records with 13 features, including demographics (age, gender, geography), account information (balance, tenure, number of products), and binary status indicators (has credit card, is active member). The target variable is 'Exited', which indicates whether a customer left the bank (1) or not (0).

This section initializes all the necessary libraries and begins the data loading and preparation process for a customer churn prediction project.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve
```

2.1. Importing Required Libraries

pandas: Used for reading and manipulating structured data (DataFrames).

numpy: Supports numerical operations, particularly with arrays and matrices.

matplotlib.pyplot and **seaborn**: Used for data visualization (e.g., correlation heatmaps, ROC curves, etc.)

2.2. Machine Learning Tools

train_test_split, **GridSearchCV**, **RandomizedSearchCV**: Used for splitting the dataset and tuning model hyperparameters.

LabelEncoder, StandardScaler: For preprocessing—encoding categorical variables and scaling features respectively.

SimpleImputer: Handles missing values if present.

SMOTE: A technique for balancing imbalanced datasets by generating synthetic samples of the minority class.

Classification model:DecisionTreeClassifier is a supervised machine learning algorithm, the model works by learning simple if-then decision rules inferred from the input data features. It builds a tree-like structure, where each internal node represents a decision based on a feature

Evaluation metrics:Include accuracy, F1 score, confusion matrix, and ROC-AUC score to assess model performance.

Plot_tree: Used for visualizing the trained Decision Tree model.

2.3.Loading Dataset

```
[ ]  # *1. Veri Yükleme ve Ön İşleme*
     df = pd.read_csv("datam.csv")
```

Loads the dataset from a CSV file into a pandas DataFrame called df.

2.4. Separating Features and Target Variable

```
[ ]  X = df.drop("Exited", axis=1)
     y = df["Exited"]
```

X: Contains all input features except the target variable.

y: Contains the target variable, **'Exited'**, which indicates whether a customer has left the bank (1) or stayed (0).

```
[ ]  X.head()
```

	id	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	0	15674932	Okwudilichukwu	668	France	Male	33.0	3	0.00	2	1.0	0.0	181449.97
1	1	15749177	Okwudiliolisa	627	France	Male	33.0	1	0.00	2	1.0	1.0	49503.50
2	2	15694510	Hsueh	678	France	Male	40.0	10	0.00	2	1.0	0.0	184866.69
3	3	15741417	Kao	581	France	Male	34.0	2	148882.54	1	1.0	1.0	84560.88
4	4	15766172	Chiemenam	716	Spain	Male	33.0	5	0.00	2	1.0	1.0	15068.83

This displays the first 5 rows of the feature dataset X, which contains all independent variables used for training the churn prediction model.

2.5.Checking Missing Values

```
[ ] X.isnull().sum().sort_values(ascending=False)
```

	0
id	0
CustomerId	0
Surname	0
CreditScore	0
Geography	0
Gender	0
Age	0
Tenure	0
Balance	0
NumOfProducts	0
HasCrCard	0
IsActiveMember	0
EstimatedSalary	0

dtype: int64

```
[ ] X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165034 entries, 0 to 165033
Data columns (total 13 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               165034 non-null  int64
 1   CustomerId       165034 non-null  int64
 2   Surname          165034 non-null  object
 3   CreditScore      165034 non-null  int64
 4   Geography        165034 non-null  object
 5   Gender           165034 non-null  object
 6   Age              165034 non-null  float64
 7   Tenure           165034 non-null  int64
 8   Balance          165034 non-null  float64
 9   NumOfProducts    165034 non-null  int64
 10  HasCrCard        165034 non-null  float64
 11  IsActiveMember   165034 non-null  float64
 12  EstimatedSalary  165034 non-null  float64
dtypes: float64(5), int64(5), object(3)
memory usage: 16.4+ MB
```

This left side code block is used to check for missing values in each column of the feature matrix X, the right side code block shows general information about values which means:

Each row shows a column name and the number of missing (null) values it contains.

All values are 0 means that there are no missing values in any of the columns in the dataset.So, it's not necessary imputation step (e.g., mean/median filling) at this point.

Details of Columns Description:

Customer ID: A unique identifier for each customer

Surname: The customer's surname or last name

Credit Score: A numerical value representing the customer's credit score

Geography: The country where the customer resides (France, Spain or Germany)

Gender: The customer's gender (Male or Female)

Age: The customer's age.

Tenure: The number of years the customer has been with the bank

Balance: The customer's account balance

NumOfProducts: The number of bank products the customer uses (savings account, credit card)

HasCrCard: Whether the customer has a credit card (1 = yes, 0 = no)

IsActiveMember: Whether the customer is an active member (1 = yes, 0 = no)

EstimatedSalary: The estimated salary of the customer

Exited: Whether the customer has churned (1 = yes, 0 = no)

2.6. Encoding Categorical Features with LabelEncoder

```
[ ]  le = LabelEncoder()
     for col in X.columns:
         if X[col].dtype == 'object':  # Sadece object kolonlara uygula
             X[col] = le.fit_transform(X[col])
```

In order to train machine learning models on categorical variables, it is necessary to convert text values into numeric representations. This is accomplished using the LabelEncoder from scikit-learn. The above code block loops through all columns in the feature matrix `X` and applies label encoding to those with the data type `object`. These typically represent string-based categorical variables such as 'Geography' and 'Gender'.

This approach replaces category names with numerical codes, enabling compatibility with machine learning algorithms. For example, values such as 'France', 'Germany', and 'Spain' in the 'Geography' column may be encoded as 0, 1, and 2 respectively. This transformation preserves the categorical nature of the variable without implying any ordinal relationship.

2.7.Correlation Analysis

```
[ ]  X_corr = X.copy()
     catcol = [col for col in X_corr.columns if X_corr[col].dtype == "object"]
     le = LabelEncoder()
     for col in catcol:
         X_corr[col] = le.fit_transform(X_corr[col])

     # Korelasyon matrisini hesaplama
     correlation_matrix = X_corr.corr()

     # Korelasyon matrisini görselleştirme
     plt.figure(figsize=(30, 30))
     sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", square=True)
     plt.title("Pearson Correlation of Features", fontsize=20)
     plt.xticks(rotation=90)
     plt.yticks(rotation=0)
     plt.show()
```

To investigate the linear relationships between the independent features, a Pearson correlation matrix was computed. After encoding, the .corr() function was used to compute the Pearson correlation coefficients between all pairs of features. These coefficients measure the

strength and direction of the linear relationships, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation).

We can observe a strong positive correlation between Age and the likelihood of churn, which indicates that as age increases, the probability of a customer exiting the bank also increases. This suggests that older individuals are more likely to churn compared to younger customers. Additionally, there is a strong negative correlation between the Number of Products and IsActiveMember features and churn, implying that more active customers or those holding more products with the bank are less likely to leave. This analysis highlights that these two variables — activity status and number of products — play a crucial role in determining customer retention and should be given significant importance in both modeling and strategic business decisions.

2.8.Reduction and Class Balancing

After encoding and correlation analysis, non-informative features such as id, CustomerId, and Surname were dropped from the feature set. These variables either uniquely identify records or hold personal information that does not contribute to the prediction task.

```
[ ]  X = X.drop(['id', 'CustomerId', 'Surname',], axis=1)
```

This code removes three columns from the feature matrix X:

- **id** and **CustomerId**: These are unique identifiers that do not carry any predictive information. Including them could lead to overfitting or irrelevant pattern detection.
- **Surname**: Although it is a categorical variable, it has no meaningful relationship with the target variable and may introduce noise.

```
[ ]  # 2. SMOTE ile sınıf dengesini ayarlama*
     smote = SMOTE(sampling_strategy="auto", random_state=42)
     X, y = smote.fit_resample(X, y)
```

SMOTE (Synthetic Minority Over-sampling Technique) is used to handle class imbalance by generating synthetic samples for the minority class (in this case, customers who churned).

This step is crucial because highly imbalanced datasets can lead to biased models that perform well on the majority class but poorly on the minority.

sampling_strategy="auto" automatically balances the classes by increasing the minority class to match the majority class size.

2.9.Data Splitting and Feature Scaling

Before training the machine learning model, the dataset must be properly partitioned to evaluate its performance on unseen data. In this step, the dataset was split into training and testing sets using the train_test_split function from the scikit-learn library. A split ratio of 60% for training and 40% for testing was applied, with a fixed random_state to ensure reproducibility.

```
[ ]  # *3. Veri bölme ve ölçekleme*


     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)
     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)
```

Once the data was split, feature scaling was performed using the StandardScaler. This scaling technique transforms the features such that they have a mean of 0 and a standard deviation of 1.

Standardization is particularly important for many machine learning models, as it brings all features to a common scale and improves model convergence and performance.

Importantly, the scaler was fitted only on the training data and then applied to both training and test sets. This prevents information leakage from the test set into the model during training, ensuring a fair evaluation of the model's generalization capability.

This preprocessing step ensures that the model is trained on appropriately scaled and separated data, laying the foundation for reliable and valid predictive performance.

2.10. Hyperparameter Tuning with RandomizedSearchCV

To optimize the performance of the Decision Tree model, RandomizedSearchCV was used to tune key hyperparameters such as max_depth, min_samples_split, min_samples_leaf, and criterion. Instead of testing all combinations, it randomly explored 10 parameter sets using 3-fold cross-validation.

```
[ ]  dt_params = {
        'max_depth': [3, 5, 7, 10, None],
        'min_samples_split': [2, 5, 10, 20],
        'min_samples_leaf': [1, 2, 5, 10],
        'criterion': ['gini', 'entropy']
    }
    # n_iter: denenmesini istediğin rastgele kombinasyon sayısı (ör: 10)
    random_search = RandomizedSearchCV(
        DecisionTreeClassifier(random_state=42),
        dt_params,
        n_iter=10,
        cv=3,
        scoring='accuracy',
        n_jobs=-1,
        random_state=42
    )
    random_search.fit(X_train, y_train)

    model = random_search.best_estimator_
    print("En iyi parametreler:", random_search.best_params_)
```

```
En iyi parametreler: {'min_samples_split': 20, 'min_samples_leaf': 2, 'max_depth': 10, 'criterion': 'gini'}
```

The best model was selected based on accuracy and included the parameters: max_depth=10, min_samples_split=20, min_samples_leaf=2, and criterion='gini'. This tuning process helped improve model accuracy while reducing the risk of overfitting.

3.Model Training and Evaluation

After completing data preprocessing and hyperparameter tuning, the optimized Decision Tree classifier was trained on the prepared training dataset using the .fit() method. Following training, predictions were made on both the training (X_train) and test (X_test) sets to assess model performance.

```
[ ]    model.fit(X_train, y_train)


       y_pred_train = model.predict(X_train)
       y_pred_test = model.predict(X_test)
```

```
[ ]  train_accuracy = accuracy_score(y_train, y_pred_train)
     test_accuracy = accuracy_score(y_test, y_pred_test)

     print("Eğitim doğruluğu:", train_accuracy)
     print("Test doğruluğu:", test_accuracy)

     print("\nEğitim Sınıflandırma Raporu:")
     print(classification_report(y_train, y_pred_train))

     print("\nTest Sınıflandırma Raporu:")
     print(classification_report(y_test, y_pred_test))
```

```
Eğitim doğruluğu: 0.8830947577416979
Test doğruluğu: 0.878106656675409

Eğitim Sınıflandırma Raporu:
              precision    recall  f1-score   support

           0       0.87      0.90      0.89     78209
           1       0.90      0.87      0.88     77926

    accuracy                           0.88    156135
   macro avg       0.88      0.88      0.88    156135
weighted avg       0.88      0.88      0.88    156135


Test Sınıflandırma Raporu:
              precision    recall  f1-score   support

           0       0.87      0.89      0.88     51904
           1       0.89      0.86      0.88     52187

    accuracy                           0.88    104091
   macro avg       0.88      0.88      0.88    104091
weighted avg       0.88      0.88      0.88    104091
```

The model achieved an accuracy of 88.3% on the training set and 87.8% on the test set, indicating that it generalizes well without overfitting. The similarity between training and test performance suggests that the model successfully captured relevant patterns in the data without learning noise.

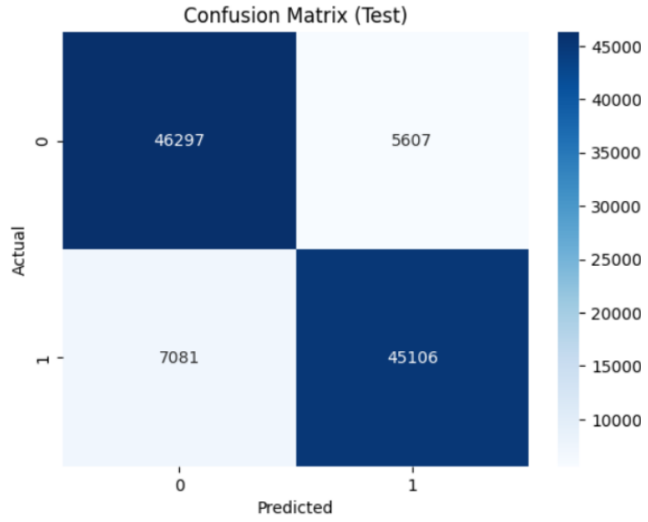
The classification reports for both sets showed the following:

-Precision, Recall, and F1-score for both classes (0: Not Exited, 1: Exited) were consistently around 0.87–0.90, reflecting a strong balance between false positives and false negatives.

-Support values for both classes were balanced due to prior SMOTE application, which helped in improving model fairness across both outcomes.

Confusion Matrix Analysis

```python
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred_test)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix (Test)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



The classification report for the test set shows balanced precision, recall, and F1-scores across both classes:

For class 0 (Not Exited): precision = 0.87, recall = 0.89, F1-score = 0.88

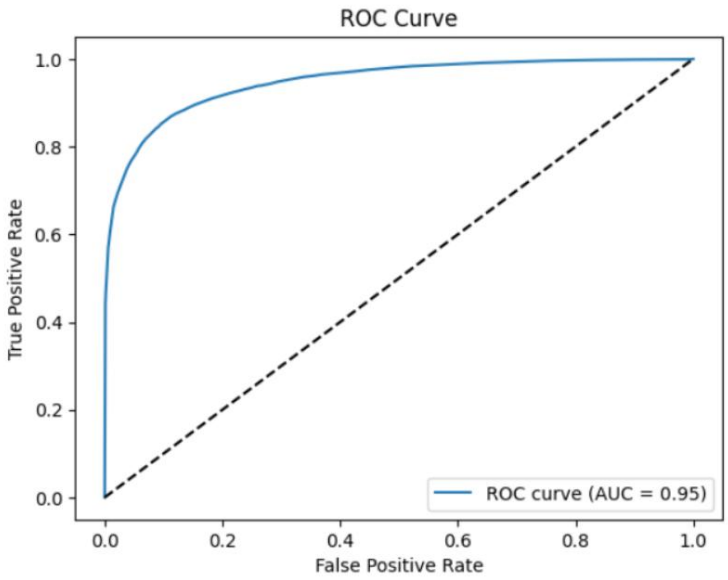
For class 1 (Exited): precision = 0.89, recall = 0.86, F1-score = 0.88

The model correctly identified 45,106 churned customers and 46,297 retained customers. Although there were some misclassifications (7,081 false negatives and 5,607 false positives), the overall performance remains strong.This demonstrates that the model performs well in identifying both retained and churned customers, with slightly stronger precision in detecting churned customers.

the overall sensitivity and specificity were well balanced. These results confirm that the Decision Tree model, trained and tuned using robust techniques, is both reliable and effective in predicting customer churn in a balanced and interpretable manner.

ROC Curve and AUC Score

```
# ROC Curve ve AUC
if len(np.unique(y_test)) == 2:
    y_proba = model.predict_proba(X_test)[:, 1]
    auc = roc_auc_score(y_test, y_proba)
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    plt.plot(fpr, tpr, label=f'ROC curve (AUC = {auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend()
    plt.show()
```



To further evaluate the model's discriminatory power, a Receiver Operating Characteristic (ROC) curve was plotted, and the Area Under the Curve (AUC) was computed. The ROC curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate at various classification thresholds.

In the graph, the x-axis represents the False Positive Rate (FPR), and the y-axis represents the True Positive Rate (TPR). The curve provides insight into the model's performance across different threshold values, offering a threshold-independent evaluation metric.

The ROC curve plots the true positive rate against the false positive rate, and the Area Under the Curve (AUC) was 0.95 which indicates excellent classification capability. An AUC score closer to 1.0 implies that the model is highly effective in distinguishing between the two classes—customers who exited and those who did not.

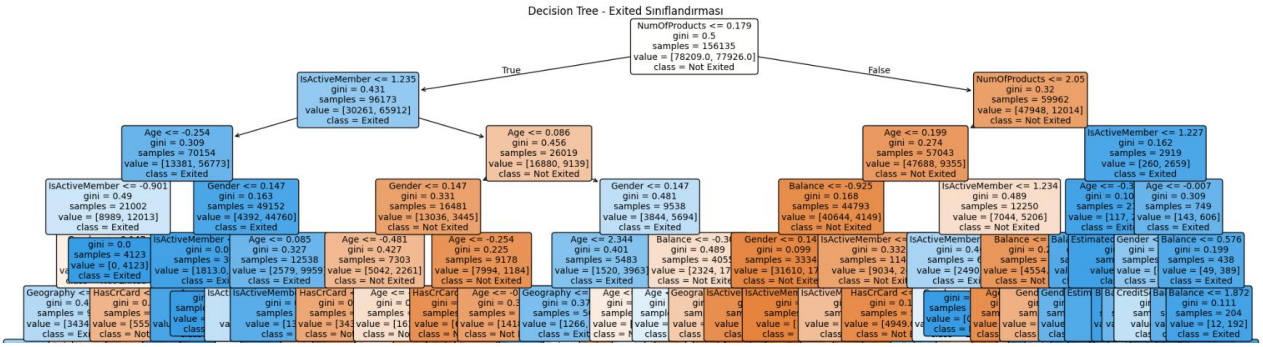
The plotted ROC curve shows a sharp rise towards the upper left corner of the plot, further confirming the model's strong performance. This implies a high true positive rate with a low false positive rate, which is particularly important in churn prediction tasks where minimizing missed churners is critical.

In summary, the ROC-AUC analysis reinforces the findings from accuracy and F1-score metrics, validating the model as a robust and reliable tool for customer churn classification.

4.Decision Tree Visualization

To enhance interpretability, the trained Decision Tree model was visualized using the plot_tree() function from the sklearn.tree module. This function generates a graphical representation of the tree structure, displaying how decisions are made based on feature values at each node.

```
[ ]  # --- Decision Tree Görselleştirmesi ---
     plt.figure(figsize=(24, 12))
     plot_tree(
         model,
         feature_names=X.columns,
         class_names=['Not Exited', 'Exited'],
         filled=True,
         rounded=True,
         fontsize=10,
         )
     plt.title('Decision Tree - Exited Sınıflandırması')
     plt.show()
```



The Decision Tree classifier, supported by effective preprocessing (e.g., SMOTE, scaling), hyperparameter tuning, and a strong feature set, produced highly reliable results. The combination of strong accuracy, balanced classification metrics, and high AUC confirms that the model is both accurate and robust in predicting customer churn.

5.Testing, Simulating and Prediction for New Customers

To simulate how the trained model would behave in a real-world deployment, a batch prediction test was conducted using manually created hypothetical customer data. A list of 10 new customer profiles was created, each defined by relevant features such as credit score, age, gender, country, account balance, tenure, and product usage.

Code:

```
# --- 10 Yeni Müşterilik Tahmin ---
customer_list = [
    {"CreditScore": 650, "Geography": "France", "Gender": "Female", "Age": 35.0, "Tenure": 5, "Balance": 15000.0, "NumOfProducts": 2, "HasCrCard": 1.0, "IsActiveMember": 1.0, "EstimatedSalary": 50000.0},
    {"CreditScore": 720, "Geography": "Germany", "Gender": "Male", "Age": 42.0, "Tenure": 8, "Balance": 0.0, "NumOfProducts": 1, "HasCrCard": 0.0, "IsActiveMember": 0.0, "EstimatedSalary": 70000.0},
    {"CreditScore": 580, "Geography": "Spain", "Gender": "Female", "Age": 29.0, "Tenure": 2, "Balance": 35000.0, "NumOfProducts": 3, "HasCrCard": 1.0, "IsActiveMember": 1.0, "EstimatedSalary": 30000.0},
    {"CreditScore": 800, "Geography": "France", "Gender": "Male", "Age": 50.0, "Tenure": 10, "Balance": 50000.0, "NumOfProducts": 2, "HasCrCard": 1.0, "IsActiveMember": 0.0, "EstimatedSalary": 120000.0},
    {"CreditScore": 430, "Geography": "Germany", "Gender": "Female", "Age": 23.0, "Tenure": 1, "Balance": 0.0, "NumOfProducts": 1, "HasCrCard": 0.0, "IsActiveMember": 1.0, "EstimatedSalary": 25000.0},
    {"CreditScore": 690, "Geography": "Spain", "Gender": "Male", "Age": 40.0, "Tenure": 6, "Balance": 20000.0, "NumOfProducts": 2, "HasCrCard": 1.0, "IsActiveMember": 1.0, "EstimatedSalary": 75000.0},
    {"CreditScore": 510, "Geography": "France", "Gender": "Female", "Age": 31.0, "Tenure": 3, "Balance": 10000.0, "NumOfProducts": 1, "HasCrCard": 1.0, "IsActiveMember": 1.0, "EstimatedSalary": 40000.0},
    {"CreditScore": 600, "Geography": "Spain", "Gender": "Male", "Age": 27.0, "Tenure": 4, "Balance": 5000.0, "NumOfProducts": 2, "HasCrCard": 0.0, "IsActiveMember": 0.0, "EstimatedSalary": 35000.0},
    {"CreditScore": 750, "Geography": "Germany", "Gender": "Male", "Age": 38.0, "Tenure": 9, "Balance": 30000.0, "NumOfProducts": 3, "HasCrCard": 1.0, "IsActiveMember": 1.0, "EstimatedSalary": 95000.0},
    {"CreditScore": 670, "Geography": "France", "Gender": "Female", "Age": 44.0, "Tenure": 7, "Balance": 25000.0, "NumOfProducts": 2, "HasCrCard": 1.0, "IsActiveMember": 1.0, "EstimatedSalary": 60000.0}
]

customer_df = pd.DataFrame(customer_list)

# Kategorik kolonları encode et (eğitimdeki encoder ile)
for col in ["Geography", "Gender"]:
    if col in customer_df.columns:
        le.fit(df[col].astype(str))
        customer_df[col] = le.transform(customer_df[col].astype(str))

customer_df = customer_df.fillna(0)
customer_df_scaled = scaler.transform(customer_df)
predictions = model.predict(customer_df_scaled)

for i, customer in enumerate(customer_list):
    print(f"Müşteri {i+1}: Tahmin = {'Exited' if predictions[i]==1 else 'Not Exited'}")
```

Output:

```
Müşteri 1: Tahmin = Not Exited
Müşteri 2: Tahmin = Exited
Müşteri 3: Tahmin = Exited
Müşteri 4: Tahmin = Not Exited
Müşteri 5: Tahmin = Not Exited
Müşteri 6: Tahmin = Not Exited
Müşteri 7: Tahmin = Not Exited
Müşteri 8: Tahmin = Not Exited
Müşteri 9: Tahmin = Exited
Müşteri 10: Tahmin = Not Exited
```

Console-Based Prediction

In order to enhance user accessibility and simulate real-time decision support, an interactive console application was implemented. This component enables a user (e.g., a bank employee or customer service agent) to input feature values for a single customer and receive an instant prediction regarding churn likelihood.

Code :

```
# --- Konsoldan Müşteri Bilgisi Alıp Tahmin Yapma (Sayısal Kategori Seçimi ile) ---
print("\nKendi müşteri bilginizi girerek tahmin alabilirsiniz!")

# Kategorik değer eşleştirmeleri
geography_map = {v: i for i, v in enumerate(df['Geography'].astype(str).unique())}
gender_map = {v: i for i, v in enumerate(df['Gender'].astype(str).unique())}

print("Geography kodları:")
for k, v in geography_map.items():
    print(f"  {k} = {v}")
print("Gender kodları:")
for k, v in gender_map.items():
    print(f"  {k} = {v}")

input_dict = {}
columns = [
    ("CreditScore", float),
    ("Geography", int),
    ("Gender", int),
    ("Age", float),
    ("Tenure", int),
    ("Balance", float),
    ("NumOfProducts", int),
    ("HasCrCard", float),
    ("IsActiveMember", float),
    ("EstimatedSalary", float)
]

for col, typ in columns:
    if col == "Geography":
        val = input(f"{col} (" + ", ".join([f'{k}={v}' for k,v in geography_map.items()]) + "): ")
    elif col == "Gender":
        val = input(f"{col} (" + ", ".join([f'{k}={v}' for k,v in gender_map.items()]) + "): ")
    else:
        val = input(f"{col}: ")
    if val.strip() == "":
        input_dict[col] = 0
    else:
        input_dict[col] = typ(val)

# Kategorik sayısal değerleri doğrudan kullan
user_df = pd.DataFrame([input_dict])
user_df_scaled = scaler.transform(user_df)
user_pred = model.predict(user_df_scaled)[0]
```

Output :

```
Kendi müşteri bilginizi girerek tahmin alabilirsiniz!
Geography kodları:
  France = 0
  Spain = 1
  Germany = 2
Gender kodları:
  Male = 0
  Female = 1
```

6. Conclusion

This project successfully developed a Decision Tree-based model to predict customer churn using structured bank data. Through careful preprocessing, class balancing with SMOTE, and hyperparameter tuning, the model achieved strong performance with a test accuracy of 87.8% and an AUC of 0.95.

The workflow included model interpretability via visualization and usability through both batch and interactive prediction interfaces. These features ensure that the model is not only accurate but also practical for real-world deployment in customer retention strategies.

Overall, the system demonstrates a reliable and explainable approach to churn prediction, offering valuable support for decision-making in customer relationship management.

Project Codes on Colab

https://colab.research.google.com/drive/1ItWERkIv9_aG0NXxuL79qRS9HrnO-_5K?usp=sharing#scrollTo=mtugDwMrTVTc

REFERENCES:

Article -1 : Podgorelec V, Kokol P, Stiglic B, Rozman I. Decision trees: an overview and their use in medicine. J Med Syst. 2002 Oct;26(5):445-63. doi: 10.1023/a:1016409317640. PMID: 12182209.

Summary and Scope: It presents a thorough overview of decision tree algorithms, with a specific focus on their applications in medical decision-making. The article discusses traditional induction methods such C4.5, emphasizing their reliance on statistical metrics like information gain and gain ratio. It compares these with other methodologies like CART, which uses the Gini index for tree construction. It also highlights newer, advanced strategies including oblique partitioning, pruning, and hybrid techniques that combine decision trees with neural networks or evolutionary algorithms. The authors evaluate multiple variations of decision tree models—ranging from classical methods to advanced dynamic discretization and ensemble learning techniques. They highlight the practical application of these models in medical domains such as diagnosis (e.g., myocardial infarction, cesarean delivery), clinical decision support.

Method: The article surveys various real-world applications of decision trees in medicine, such as:Supporting early diagnosis of myocardial infarction, Enhancing accuracy in identifying

adverse drug reactions, Predicting cesarean deliveries,Integrating ICU signal data to reduce false alarms.It was shown that evolutionary and hybrid methods often outperform traditional decision trees in terms of accuracy, flexibility, and robustness—particularly in complex and noisy medical datasets.

Conclusion: Decision trees are effective, interpretable, and reliable tools in medical decision-making. Despite some limitations (e.g., sensitivity to noisy or missing data), they offer high classification accuracy and transparent logic. The study recommends combining multiple induction methods and hybrid models to improve performance and adaptability in practical applications.

Article-2 : Song YY, Lu Y. Decision tree methods: applications for classification and prediction. Shanghai Arch Psychiatry. 2015 Apr 25;27(2):130-5. doi: 10.11919/j.issn.1002-0829.215044. PMID: 26120265; PMCID: PMC4466856.

Summary and Scope : This article provides a comprehensive overview of decision tree methods and their use in classification and prediction tasks, particularly in medical and health-related research. Decision trees are emphasized for their interpretability, ability to handle both numerical and categorical data, and robustness in the presence of missing values. Several algorithms are compared, including CART, C4.5, CHAID, and QUEST, with details on their splitting criteria, pruning strategies, and suitable data types.

Method: An application example is presented from a four-year cohort study examining the risk of Major Depressive Disorder (MDD). Using decision tree analysis, the authors identified subgroups with significantly elevated risk, such as unemployed male smokers, with risk levels.

Conclusion : The article concludes that decision trees are powerful and interpretable tools that can be effectively used for variable selection, risk stratification, and predictive modeling—especially in healthcare domains. However, they require careful tuning to avoid overfitting and misinterpretation.

Article- 3 : Blockeel H, Devos L, Frénay B, Nanfack G, Nijssen S. Decision trees: from efficient prediction to responsible AI. Front Artif Intell. 2023 Jul 26;6:1124553. doi: 10.3389/frai.2023.1124553. PMID: 37565044; PMCID: PMC10411911.

Summary and Scope : This article explores the evolution of decision tree algorithms, emphasizing their transformation from basic predictive models into critical components of responsible and explainable AI (XAI) systems. The authors highlight how decision trees are uniquely positioned to address ethical and interpretability challenges in high-stakes domains such as medicine, healthcare, and justice.

Method : Introduces the appeal of decision trees — their simplicity, speed, and transparency especially when explainability is crucial.Reviews classical decision tree algorithms (ID3, C4.5, CART), discussing how they partition feature space and perform pruning. Discusses modern variations, Analyzes decision trees through the lens of responsible AI, focusing on fairness, bias detection, and the importance of transparency in societal decision-making.Presents examples of medical applications, where tree-based models are preferred for their clarity, especially in clinical decision support and diagnostics

.**Conclusion :** Decision trees are not only efficient classifiers but also powerful tools for enabling ethical, explainable, and responsible AI. Their ability to expose decision logic makes them crucial in domains where trust and fairness are essential. The article argues that enhancing decision trees with modern techniques ensures they remain competitive while satisfying the transparency demands of responsible AI systems.