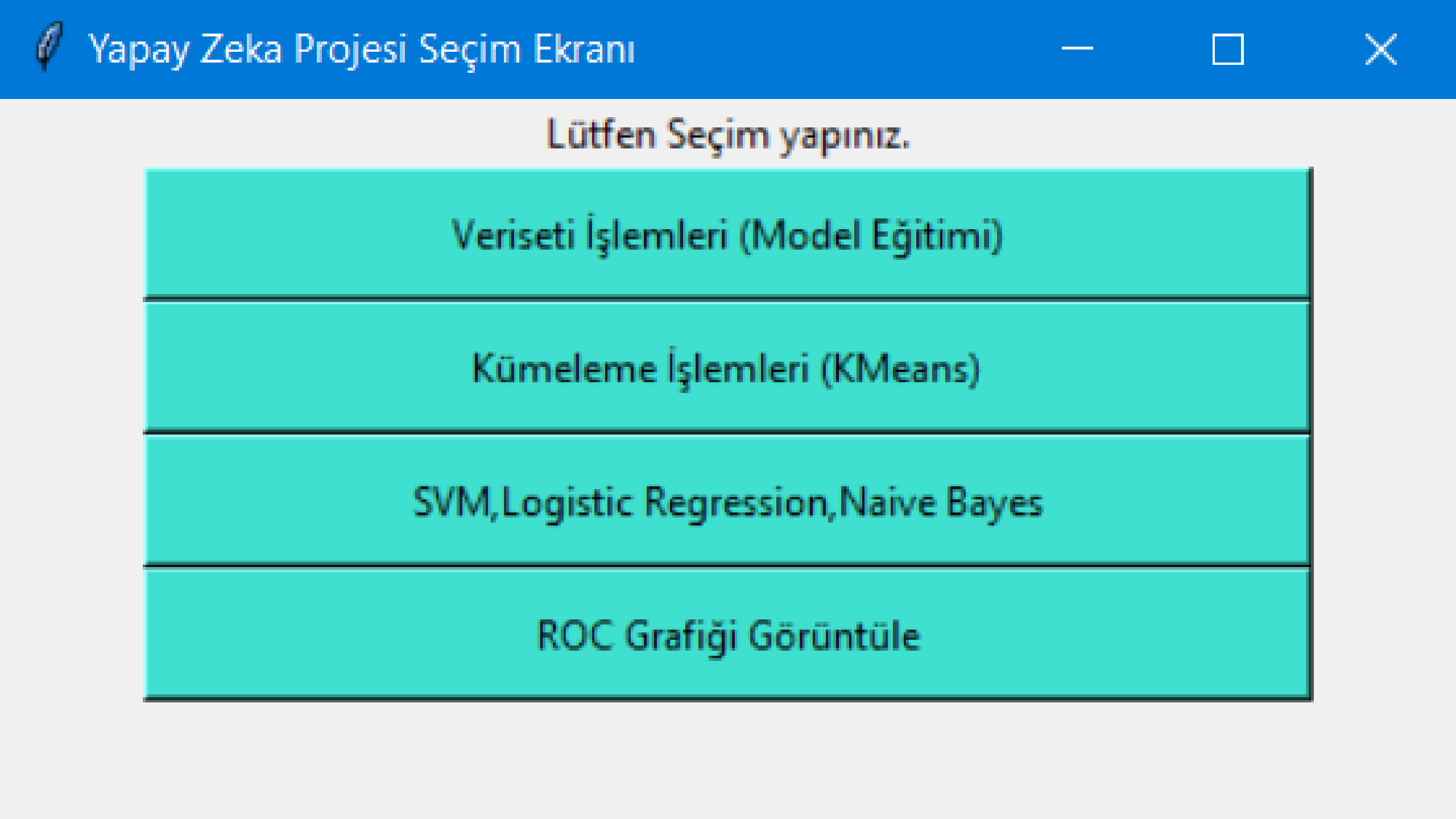
İnfluencer Effect

ALPER SARGıN

MEHMET AKIF ALTıNSOY

Main.py

```
Click here to ask Blackbox to help you code faster
import tkinter as tk
from tkinter import ttk
from veriseti import Veriseti
from kumeleme import Kumeleme
from svm_log_naive import Svm_log_naive
from roc import Roc

class SecimEkrani:
    def __init__(self, master):
        self.master = master
        self.master.title("Yapay Zeka Projesi Seçim Ekranı")
        self.master.geometry("450x250")

        tk.Label(self.master, text="Lütfen Seçim yapınız.").pack()

        tk.Button(self.master, text="Veriseti İşlemleri (Model Eğitimi)", bg="turquoise", height="2", width="50", command=self.handle_veriseti).pack()
        tk.Button(self.master, text="Kümeleme İşlemleri (KMeans)", bg="turquoise", height="2", width="50", command=self.handle_kumeleme).pack()
        tk.Button(self.master, text="SVM,Logistic Regression,Naive Bayes", bg="turquoise", height="2", width="50", command=self.handle_svm_log_naive).pack()
        tk.Button(self.master, text="ROC Grafiği Görüntüle", bg="turquoise", height="2", width="50", command=self.handle_roc).pack()

    def handle_veriseti(self):
        Veriseti()

    def handle_kumeleme(self):
        Kumeleme()

    def handle_svm_log_naive(self):
        Svm_log_naive()

    def handle_roc(self):
        Roc()

def main():
    root = tk.Tk()
    secim_ekrani = SecimEkrani(root)
    root.mainloop()
```


Yapay Zeka Projesi Seçim Ekranı
Lütfen Seçim yapınız.
Veriseti İşlemleri (Model Eğitimi)
Kümeleme İşlemleri (KMeans)
SVM,Logistic Regression,Naive Bayes
ROC Grafiği Görüntüle

Veriseti İşlemleri ve Model Eğitimi

Click here to ask Blackbox to help you code faster

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

def Veriseti():
    data = pd.read_csv('PeerIndex.csv')

    X = data.drop('Choice', axis=1)
    y = data['Choice']

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

    model = LogisticRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    print("\n")
    print("Logistic Regression")
    print("Doğruluk Oranı:", accuracy_score(y_test, y_pred))
    print("Sınıflandırma Raporu:\n", classification_report(y_test, y_pred))


    model = RandomForestClassifier()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    print("\n")
    print("RFC")
    print("Doğruluk Oranı:", accuracy_score(y_test, y_pred))
    print("Sınıflandırma Raporu:\n", classification_report(y_test, y_pred))
```

```
Logistic Regression
Doğruluk Oranı: 0.7427272727272727
Sınıflandırma Raporu:
              precision    recall  f1-score   support

           0       0.73      0.73      0.73       530
           1       0.75      0.75      0.75       570

    accuracy                           0.74      1100
   macro avg       0.74      0.74      0.74      1100
weighted avg       0.74      0.74      0.74      1100



RFC
Doğruluk Oranı: 0.7681818181818182
Sınıflandırma Raporu:
              precision    recall  f1-score   support

           0       0.75      0.78      0.76       530
           1       0.79      0.76      0.77       570

    accuracy                           0.77      1100
   macro avg       0.77      0.77      0.77      1100
weighted avg       0.77      0.77      0.77      1100
```

Kümeleme İşlemleri

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV


def Kumeleme():
    data = pd.read_csv('PeerIndex.csv')
    X = data.drop('Choice', axis=1)
    y = data['Choice']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    param_grid = {'n_estimators': [50, 100, 200],
                  'max_depth': [None, 10, 20]}
    grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
    grid_search.fit(X_train, y_train)

    X_cluster = X[['A_follower_count', 'A_following_count', 'A_listed_count', 'A_mentions_received', 'A_retweets_received', 'A_mentions_sent',
    'A_retweets_sent', 'A_posts', 'A_network_feature_1', 'A_network_feature_2', 'A_network_feature_3']]

    kmeans = KMeans(n_clusters=2, random_state=42)
    X['cluster'] = kmeans.fit_predict(X_cluster)

    plt.figure(figsize=(12, 8))
    sns.scatterplot(x='A_network_feature_1', y='A_network_feature_2', hue='cluster', data=X, palette='Set1', marker='X', s=150, edgecolor='black', linewidth=0.
    8)
    sns.scatterplot(x='A_network_feature_1', y='A_network_feature_2', hue=y_train, data=X_train, palette='viridis', alpha=0.6, s=80, edgecolor='w', linewidth=0.
    5)
    plt.title('Kümeleme Sonuçları ve Sınıflar')
    plt.show()
```

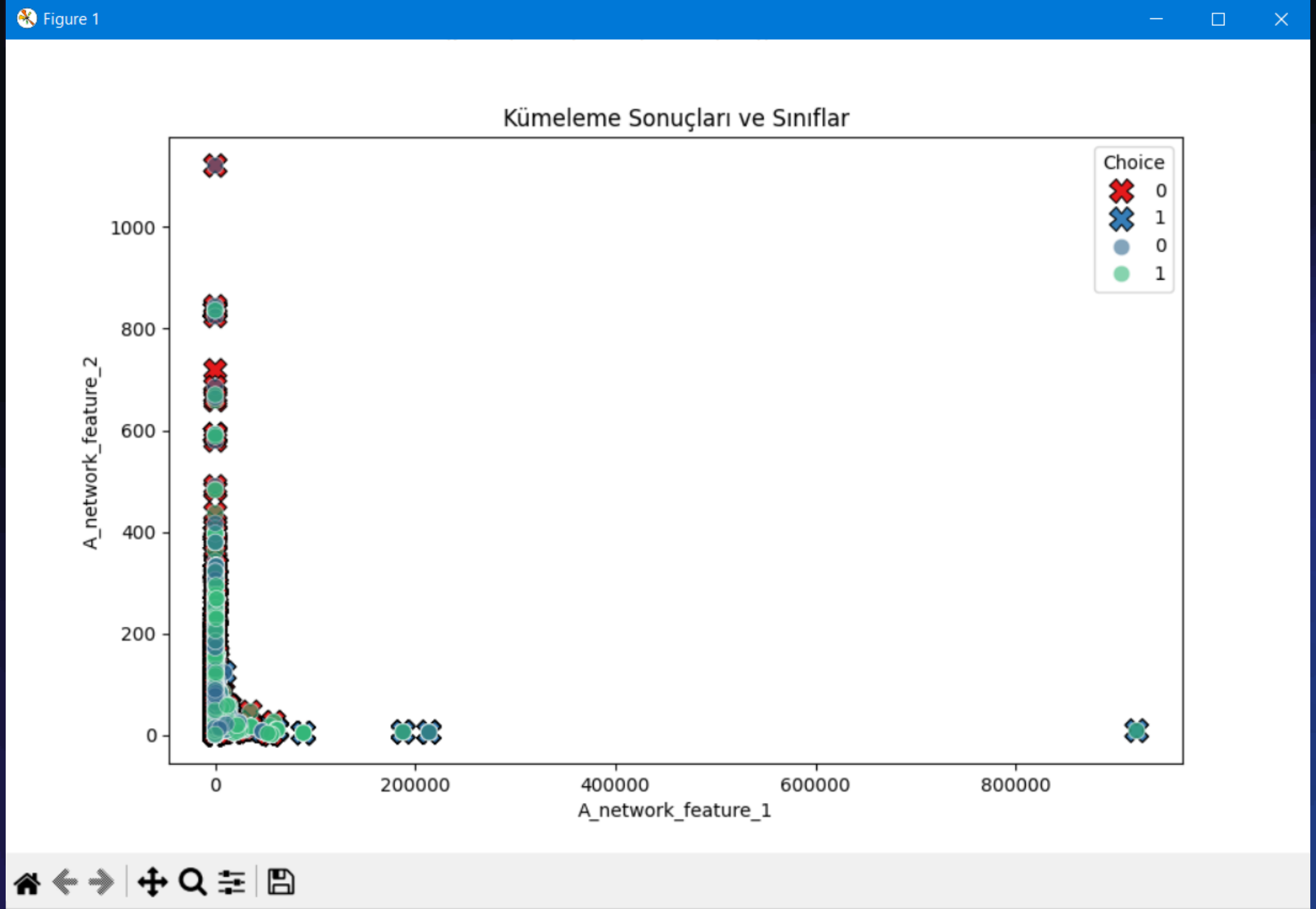
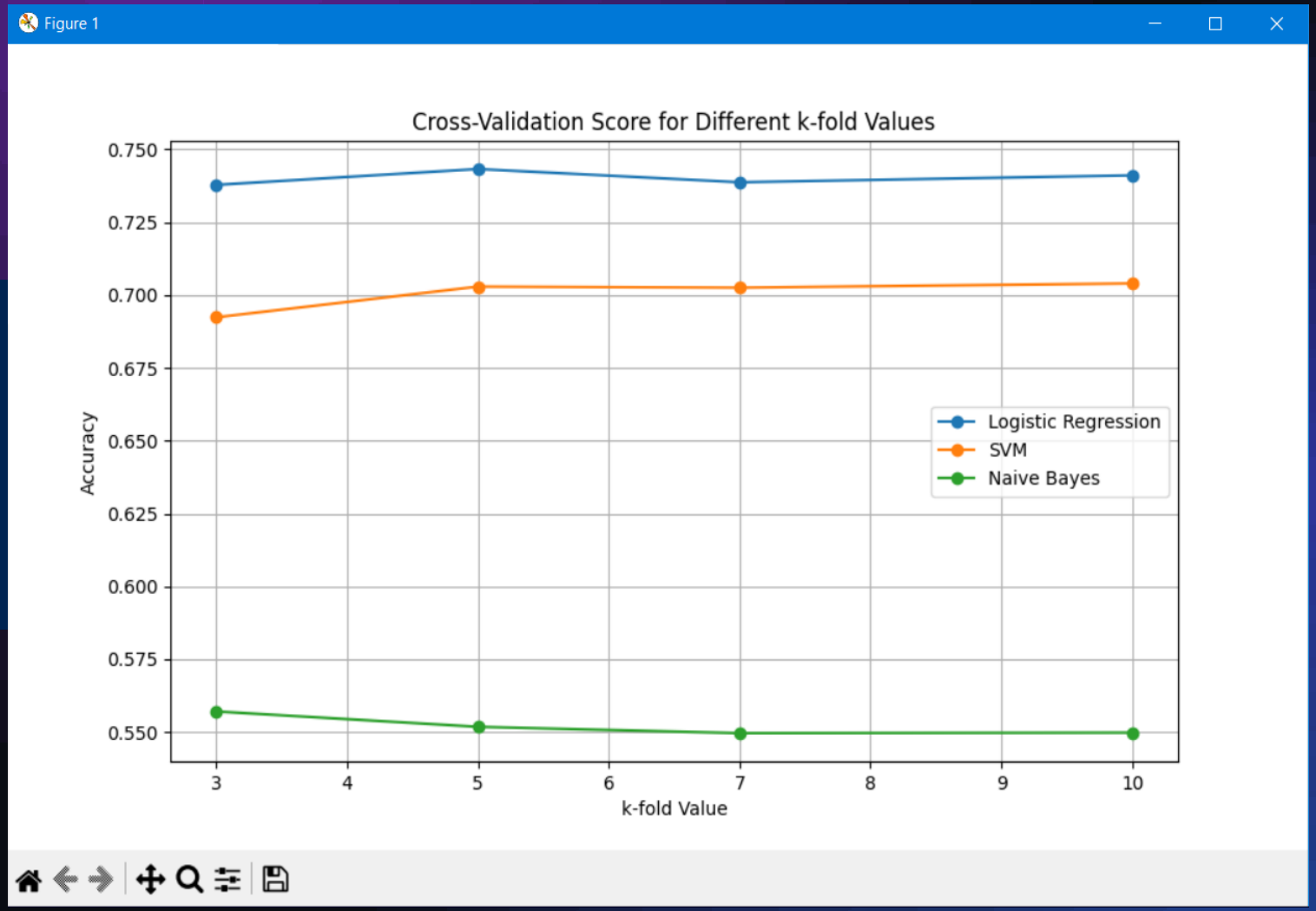


Figure 1
Kümeleme Sonuçları ve Sınıflar
Choice
0
1
0
1
A_network_feature_2
1000
800
600
400
200
0
0
200000
400000
600000
800000
A_network_feature_1

Destek Vektör Makineleri (SVM),Logistic Regresyon,Naive Bayes

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
import pandas as pd

veri = pd.read_csv('PeerIndex.csv')
X = veri.drop('Choice', axis=1)
y = veri['Choice']

kfold_degerleri = [3, 5, 7, 10]

cv_skorlari_lr = []
for kfold in kfold_degerleri:
    lr_modeli = LogisticRegression()
    skorlar = cross_val_score(lr_modeli, X, y, cv=kfold)
    ortalama_skor = np.mean(skorlar)
    cv_skorlari_lr.append(ortalama_skor)
    print(f"Lojistik Regresyon, k-fold={kfold} - Ortalama Doğruluk: {ortalama_skor:.4f}")

cv_skorlari_svm = []
for kfold in kfold_degerleri:
    svm_modeli = SVC()
    skorlar = cross_val_score(svm_modeli, X, y, cv=kfold)
    ortalama_skor = np.mean(skorlar)
    cv_skorlari_svm.append(ortalama_skor)
    print(f"SVM, k-fold={kfold} - Ortalama Doğruluk: {ortalama_skor:.4f}")

cv_skorlari_nb = []
for kfold in kfold_degerleri:
    nb_modeli = GaussianNB()
    skorlar = cross_val_score(nb_modeli, X, y, cv=kfold)
    ortalama_skor = np.mean(skorlar)
    cv_skorlari_nb.append(ortalama_skor)
    print(f"Naive Bayes, k-fold={kfold} - Ortalama Doğruluk: {ortalama_skor:.4f}")

plt.figure(figsize=(10, 6))
plt.plot(kfold_degerleri, cv_skorlari_lr, marker='o', linestyle='-', label='Lojistik Regresyon')
plt.plot(kfold_degerleri, cv_skorlari_svm, marker='o', linestyle='-', label='SVM')
plt.plot(kfold_degerleri, cv_skorlari_nb, marker='o', linestyle='-', label='Naive Bayes')
plt.title('Farklı k-fold Değerleri için Çapraz Doğrulama Skorları')
plt.xlabel('k-fold Değeri')
plt.ylabel('Doğruluk')
plt.legend()
plt.grid(True)
plt.show()
```

```
Lojistik Regresyon, k-fold=10 - Ortalama Doğruluk: 0.7411
SVM, k-fold=3 - Ortalama Doğruluk: 0.6924
SVM, k-fold=5 - Ortalama Doğruluk: 0.7029
SVM, k-fold=7 - Ortalama Doğruluk: 0.7025
SVM, k-fold=10 - Ortalama Doğruluk: 0.7040
Naive Bayes, k-fold=3 - Ortalama Doğruluk: 0.5571
Naive Bayes, k-fold=5 - Ortalama Doğruluk: 0.5518
Naive Bayes, k-fold=7 - Ortalama Doğruluk: 0.5496
Naive Bayes, k-fold=10 - Ortalama Doğruluk: 0.5498
```



ROC GRAFİĞİ

```python
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
import pandas as pd

def Roc():
    veri = pd.read_csv('PeerIndex.csv')
    X = veri.drop('Choice', axis=1)
    y = veri['Choice']

    y_bin = label_binarize(y, classes=[0, 1, 2])
    X_egitim, X_test, y_egitim, y_test = train_test_split(X, y_bin, test_size=0.2, random_state=42)

    lr_model = OneVsRestClassifier(LogisticRegression())
    lr_model.fit(X_egitim, y_egitim)
    y_skor = lr_model.decision_function(X_test)

    plt.figure(figsize=(10, 6))
    for i in range(y_bin.shape[1]):
        fpr, tpr, _ = roc_curve(y_test[:, i], y_skor[:, i])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label=f'Sınıf {i} (AUC = {roc_auc:.2f})')

    plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Rastgele')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Lojistik Regresyon için ROC Eğrisi')
    plt.legend()
    plt.grid(True)
    plt.show()
```

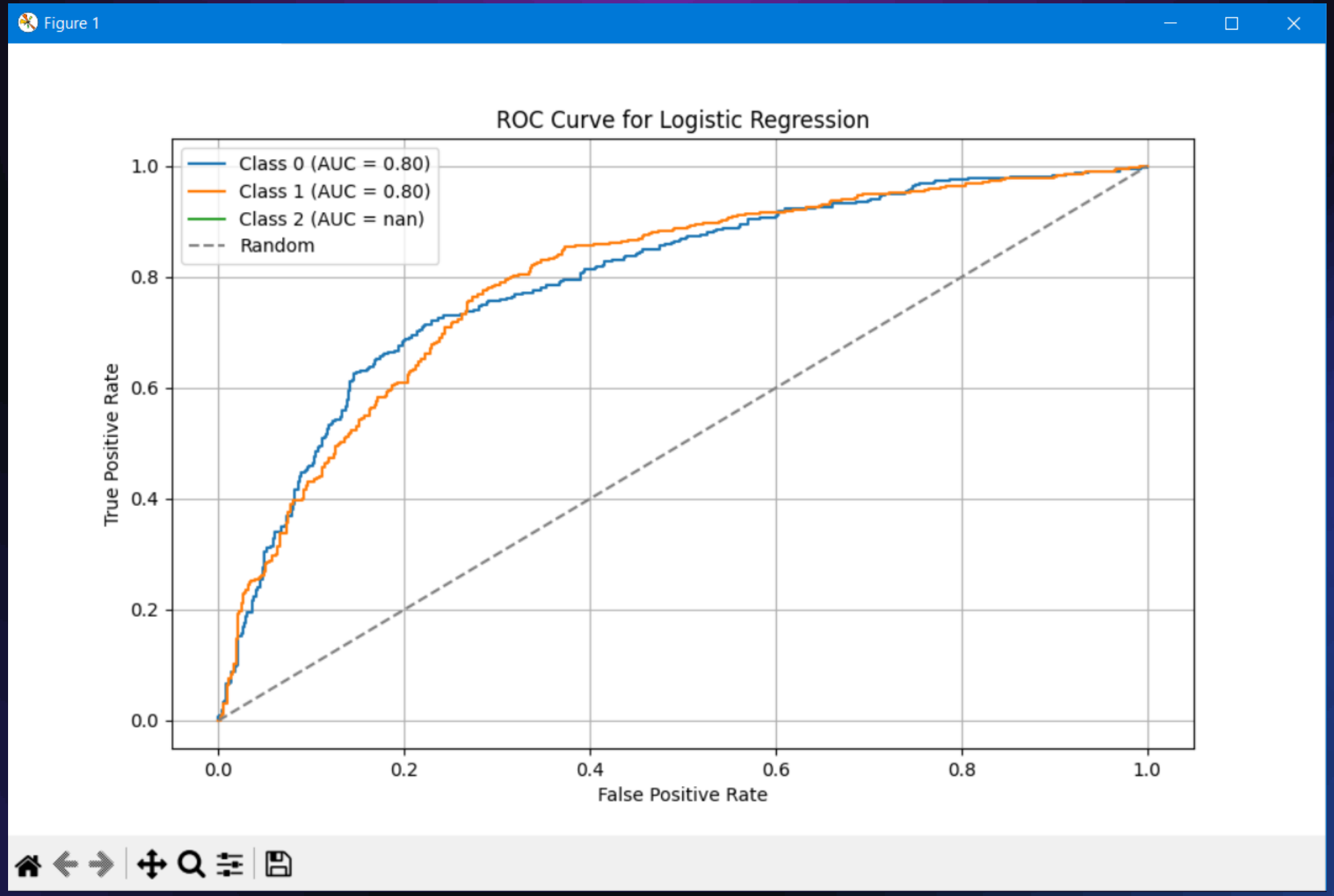


Figure 1
ROC Curve for Logistic Regression
Class 0 (AUC = 0.80)
Class 1 (AUC = 0.80)
Class 2 (AUC = nan)
Random
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Bizi Dinlediğiniz İçin Teşekkürler