Taupai Ayazhan
21B030925

Assignment 2, Data Mining

Put all deliverables into github repository in your profile. Share link to google form 24 hours before defense. Defend by explaining deliverables and answering questions.
Deliverables: .ipynb
Google form: https://docs.google.com/forms/d/e/1FAIpQLSe0GyNdOYlvM1tX_I_CtlPod5jBf-ACLGdHYZq1gVZbUeBzIg/viewform?usp=sf_link

Exercise 1: Feature Selection with `SelectKBest`

Objective: Use `SelectKBest` from scikit-learn to select the top k features from a dataset.

1. Load the Iris dataset from scikit-learn.
2. Split the dataset into features and target variable.
3. Use `SelectKBest` with the `chi2` score function to select the top 2 features.
4. Print the selected feature names.

In [1]:
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest, chi2

iris = load_iris()
X = iris.data
y = iris.target

df_X = pd.DataFrame(X, columns=iris.feature_names)

s = SelectKBest(score_func=chi2, k=2)
X_new = s.fit_transform(X, y)

mask = s.get_support()

selected_features = df_X.columns[mask]
print("Выбранные признаки:", selected_features)
```

```
Выбранные признаки: Index(['petal length (cm)', 'petal width (cm)'], dtype='object')
```

Exercise 2: Feature Importance with Random Forest

Objective: Use a Random Forest classifier to determine feature importance.

1. Load the Wine dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a Random Forest classifier on the training data.
4. Extract and visualize feature importances.

In [1]:
```
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import numpy as np

data = load_wine()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

print(X.head())
print()
print(X.info())
```

```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80
1    13.20        1.78  2.14               11.2      100.0           2.65
2    13.16        2.36  2.67               18.6      101.0           2.80
3    14.37        1.95  2.50               16.8      113.0           3.85
4    13.24        2.59  2.87               21.0      118.0           2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

   od280/od315_of_diluted_wines  proline
0                          3.92   1065.0
1                          3.40   1050.0
2                          3.17   1185.0
3                          3.45   1480.0
4                          2.93    735.0

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
dtypes: float64(13)
memory usage: 18.2 KB
None
```

In [2]:
```
X1, X2, y1, y2 = train_test_split(X, y, test_size=0.3, random_state=42)

print(f"Train sets: {X1.shape}")
print(f"Test sets: {X2.shape}")
```

```
Train sets: (124, 13)
Test sets: (54, 13)
```

In [4]:
```
# RandomForest
rf = RandomForestClassifier(random_state=42)
rf.fit(X1, y1)
```

Out[4]:
```
RandomForestClassifier(random_state=42)
```

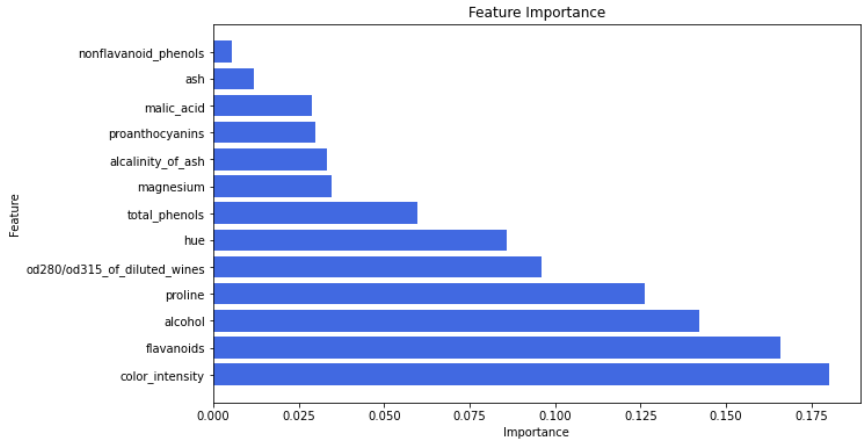
In [8]:
```
importances = rf.feature_importances_ # Извлечение важности признаков

fi_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
})

fi_df = fi_df.sort_values(by='Importance', ascending=False)
print(fi_df)
```

```
                        Feature  Importance
9                color_intensity    0.180198
6                     flavanoids    0.165947
0                        alcohol    0.142041
12                       proline    0.126080
11  od280/od315_of_diluted_wines    0.096032
10                           hue    0.085966
5                  total_phenols    0.059758
4                      magnesium    0.034673
3              alcalinity_of_ash    0.033211
8                proanthocyanins    0.029778
1                     malic_acid    0.028976
2                            ash    0.011859
7           nonflavanoid_phenols    0.005480
```

In [10]:
```
plt.figure(figsize=(10, 6))
plt.barh(fi_df['Feature'], fi_df['Importance'], color='royalblue')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importance')
plt.show()
```



Exercise 3: Recursive Feature Elimination (RFE)

Objective: Use Recursive Feature Elimination (RFE) to select features and evaluate model performance.

1. Load the Breast Cancer dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Use RFE with a Support Vector Machine (SVM) classifier to select features.
4. Train an SVM model with the selected features and evaluate its performance.

In [1]:
```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.feature_selection import RFE
from sklearn.metrics import accuracy_score

data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

svm_classifier = SVC(kernel="linear")
rfe = RFE(estimator=svm_classifier, n_features_to_select=10)
rfe.fit(X1, y1)

selected_features = X.columns[rfe.support_]
print("Выбранные признаки:")
print(selected_features)
print()

X1_rfe = rfe.transform(X1)
X2_rfe = rfe.transform(X2)

svm_classifier.fit(X1_rfe, y1)

y_pred = svm_classifier.predict(X2_rfe)
accuracy = accuracy_score(y2, y_pred)

print(f"Точность модели: {accuracy:.2f}")
```

```
Выбранные признаки:
Index(['mean radius', 'mean concavity', 'mean concave points', 'texture error',
       'worst radius', 'worst smoothness', 'worst compactness',
       'worst concavity', 'worst concave points', 'worst symmetry'],
      dtype='object')

Точность модели: 0.97
```

Exercise 4: L1 Regularization for Feature Selection

Objective: Use L1 regularization (Lasso) for feature selection.

1. Load the Diabetes dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Apply Lasso regression for feature selection.
4. Train a model using selected features and evaluate its performance.

In [2]:
```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score

data = load_diabetes()
X = data.data
y = data.target

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

l = Lasso(alpha=0.1)
l.fit(X1, y1)
print(l.coef_)
```

```
[   0.         -152.66706552  552.6941724   303.37055083  -81.3648345
   -0.         -229.25829816    0.          447.91818931   29.64235375]
```

In [3]:
```
y_pred = l.predict(X2)

mse = mean_squared_error(y2, y_pred)
r2 = r2_score(y2, y_pred)

print(f"Средняя квадратичная ошибка: {mse:.2f}")
print(f"R^2: {r2:.2f}")
```

```
Средняя квадратичная ошибка: 2798.19
R^2: 0.47
```

Classification Exercises

Exercise 1: Logistic Regression

Objective: Build a logistic regression model to classify data.

1. Load the Iris dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a logistic regression model on the training set.
4. Evaluate the model's performance on the test set using accuracy and a confusion matrix.

In [1]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

iris = load_iris()
X = iris.data
y = iris.target

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

l = LogisticRegression(max_iter=200)
l.fit(X1, y1)

y_pred = l.predict(X2)

result = accuracy_score(y2, y_pred)
print(f"Точность модели: {result:.2f}")

matrix = confusion_matrix(y2, y_pred)
print("Матрица ошибок:")
print(matrix)
```

```
Точность модели: 1.00
Матрица ошибок:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Exercise 2: Support Vector Machine (SVM)

Objective: Use an SVM classifier to classify data.

1. Load the Breast Cancer dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train an SVM model on the training data.
4. Evaluate the model's performance on the test data using accuracy and a confusion matrix.

In [1]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix


data = load_breast_cancer()
X = data.data
y = data.target

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

svm_model = SVC()
svm_model.fit(X1, y1)

y_pred = svm_model.predict(X2)


accuracy = accuracy_score(y2, y_pred)
print(f"Точность модели: {accuracy:.2f}")
print()


conf_matrix = confusion_matrix(y2, y_pred)
print("Матрица ошибок:")
print(conf_matrix)
```

```
Точность модели: 0.95

Матрица ошибок:
[[37  6]
 [ 0 71]]
```

Exercise 3: Decision Tree Classifier

Objective: Build a decision tree classifier and visualize it.

1. Load the Wine dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a decision tree classifier on the training set.

4. Visualize the decision tree.

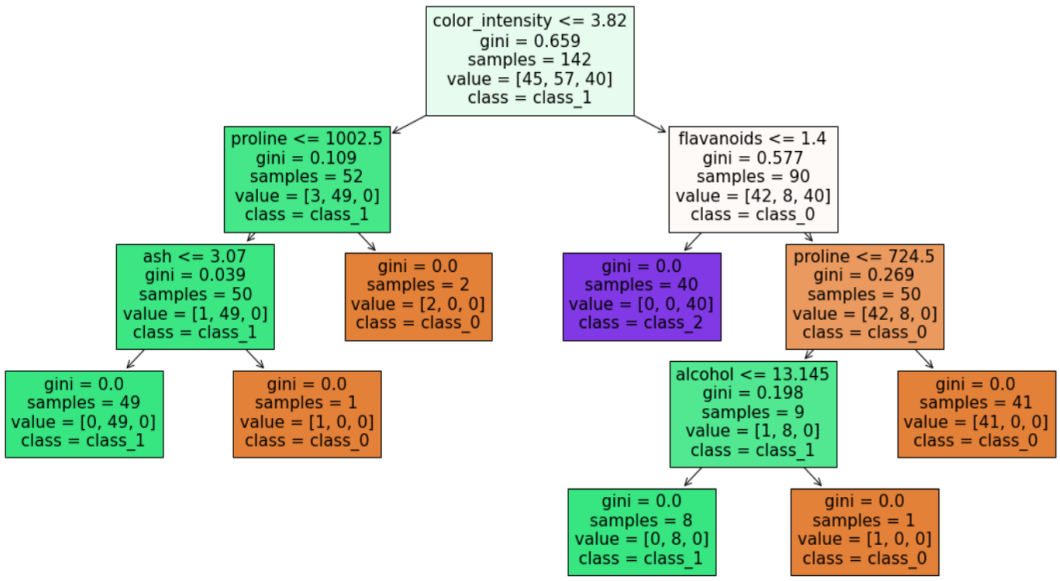
```
In [1]: from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt

data = load_wine()
X = data.data
y = data.target

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

dtc = DecisionTreeClassifier(random_state=42)
dtc.fit(X1, y1)

plt.figure(figsize=(20,10))
tree.plot_tree(dtc, feature_names=data.feature_names, class_names=data.target_names, filled=True)
plt.show()
```



Regression Exercises

Exercise 1: Linear Regression

Objective: Build a linear regression model to predict a continuous target variable.

1. Load the Boston Housing dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a linear regression model on the training set.

4. Evaluate the model's performance using mean squared error (MSE) and R-squared score.

In [2]:
```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = fetch_california_housing()
X = data.data
y = data.target

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

lr = LinearRegression()
lr.fit(X1, y1)

y_pred = lr.predict(X2)

mse = mean_squared_error(y2, y_pred)
print(f"(MSE): {mse:.2f}")

r2 = r2_score(y2, y_pred)
print(f"(R²): {r2:.2f}")
```

```
(MSE): 0.56
(R²): 0.58
```

Exercise 2: Ridge Regression

Objective: Use Ridge regression to perform regularized linear regression.

1. Load the Diabetes dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a Ridge regression model on the training set.
4. Evaluate the model's performance using mean squared error (MSE) and R-squared score.

In [1]:
```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score

data = load_diabetes()
X = data.data
y = data.target

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

ridge = Ridge(alpha=1.0)
ridge.fit(X1, y1)

y_pred = ridge.predict(X2)

mse = mean_squared_error(y2, y_pred)
r2 = r2_score(y2, y_pred)

print(f"(MSE): {mse:.2f}")
print(f"(R²): {r2:.2f}")
```

```
(MSE): 3077.41
(R²): 0.42
```

Exercise 3: Decision Tree Regression

Objective: Build a decision tree regression model and visualize it.

1. Load the Boston Housing dataset from scikit-learn.

2. Split the dataset into training and testing sets.
3. Train a decision tree regressor on the training set.
4. Evaluate the model's performance using mean squared error (MSE).
5. Visualize the decision tree.

In [1]:
```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

data = fetch_california_housing()
X = data.data
y = data.target

X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2, random_state=42)

tree = DecisionTreeRegressor(random_state=42)
tree.fit(X1, y1)

y_pred = tree.predict(X2)
mse = mean_squared_error(y2, y_pred)
print(f"(MSE): {mse:.2f}")


plt.figure(figsize=(20, 10))
plot_tree(tree, filled=True, feature_names=data.feature_names, rounded=True, max_depth=3)
plt.show()
```

```
(MSE): 0.50
```