lab09

April 29, 2024

[ ]:

```

```

[ ]:

```

```

Gradient Boosting

[ ]:

```
# Import models and utility functions
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_digits

# Setting SEED for reproducibility
SEED = 23

# Importing the dataset
X, y = load_digits(return_X_y=True)

# Splitting dataset
train_X, test_X, train_y, test_y = train_test_split(X, y,

  ↪= 0.25,

  ↪= SEED)

# Instantiate Gradient Boosting Regressor
gbc = GradientBoostingClassifier(n_estimators=300,
                                                                learning_rate=0.
  ↪05,
                                                                random_state=100,
                                                                max_features=5 )
# Fit to training set
gbc.fit(train_X, train_y)

# Predict on test set
pred_y = gbc.predict(test_X)
```

```
# accuracy
acc = accuracy_score(test_y, pred_y)
print("Gradient Boosting Classifier accuracy is : {:.2f}".format(acc))
```

```
Gradient Boosting Classifier accuracy is : 0.98
```

[ ]:
```
# Importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error

# Load the dataset
df = pd.read_csv('For_modeling.csv')

# Assuming your dataset has features and target variable
# Features are stored in X, and the target variable (trip duration) is stored␣
 ↪in y
X = df.drop(columns=['Duration'])
y = df['Duration']
# Dropping rows with missing values
X.dropna(inplace=True)
y = y[X.index]  # Align y with the updated indices of X

# Splitting 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)

# Creating the Gradient Boosting Regressor model
gb_model = GradientBoostingRegressor()

# Training the model
gb_model.fit(X_train, y_train)

# Predicting on the test set
y_pred = gb_model.predict(X_test)

# Calculating mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

[ ]:
```
# Importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
```

```
# Load the dataset
df = pd.read_csv('For_modeling.csv')

# Assuming your dataset has features and target variable
# Features are stored in X, and the target variable (trip duration) is stored␣
 ↪in y
X = df.drop(columns=['Duration'])
y = df['Duration']
# Dropping rows with missing values
X.dropna(inplace=True)
y = y[X.index]  # Align y with the updated indices of X

# Splitting 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)

# Creating the Gradient Boosting Regressor model
gbc = GradientBoostingClassifier(n_estimators=300,
                                                                 learning_rate=0.
 ↪05,
                                                                 random_state=100,
                                                                 max_features=5 )
# Fit to training set
gbc.fit(train_X, train_y)

# Predict on test set
pred_y = gbc.predict(test_X)

# accuracy
acc = accuracy_score(test_y, pred_y)
print("Gradient Boosting Classifier accuracy is : {:.2f}".format(acc))

# Calculating mean squared error
mse = mean_squared_error(test_y, pred_y)
print("Mean Squared Error:", mse)
```

```
Gradient Boosting Classifier accuracy is : 0.98
Mean Squared Error: 0.3088888888888889
```

[ ]:
```
# Dropping rows with missing values
X.dropna(inplace=True)
y = y[X.index]  # Align y with the updated indices of X
```

[ ]:
```

```

XGBoosting

```
[ ]:
```

```
[ ]: import xgboost as xgb
     # Splitting 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)

     # Creating the XGBoost model
     xgb_model = xgb.XGBRegressor()

     # Training the model
     xgb_model.fit(X_train, y_train)

     # Predicting on the test set
     y_pred = xgb_model.predict(X_test)

     # Calculating mean squared error
     mse = mean_squared_error(y_test, y_pred)
     print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 32.284545956720656
```