Name: Annie Mahajan
Registration No.: 23BRS1121

Lab - 05

Sample 1

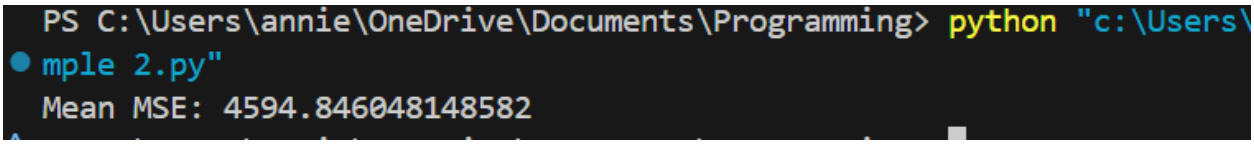
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import KFold, cross_val_score
import numpy as np
import matplotlib.pyplot as plt
# Load dataset
iris = load_iris()
X = iris.data
y = iris.target
# Create decision tree classifier
dt = DecisionTreeClassifier(criterion='gini')
# K-fold setup
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
# Perform cross validation
scores = cross_val_score(dt, X, y, cv=kfold, scoring="accuracy")
print("Accuracy scores:", scores)
print("Mean accuracy:", scores.mean())
print("Standard deviation:", scores.std())
# Fit the model on the full dataset
dt.fit(X, y)
# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(dt, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)
plt.savefig('decision_tree.png')
print("Decision tree saved as 'decision_tree.png'")
```

```
PS C:\Users\annie\OneDrive\Documents\Programming> python "c:\Users\annie\OneDrive\
mple1.py"
Accuracy scores: [1.         1.         0.93333333 0.93333333 0.93333333]
Mean accuracy: 0.9600000000000002
Standard deviation: 0.03265986323710904
```

Sample 2

```
from sklearn.datasets import load_diabetes
from sklearn.tree import DecisionTreeRegressor, plot_tree
```

```
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
import numpy as np
# Load dataset
data = load_diabetes()
X = data.data
y = data.target
# Train model
model = DecisionTreeRegressor(max_depth=5, random_state=42)
model.fit(X, y)
# Cross validation MSE
scores = cross_val_score(model, X, y, cv=5, scoring="neg_mean_squared_error")
mse = -scores.mean()
print("Mean MSE:", mse)
```



Sample 3

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import GridSearchCV, KFold
import matplotlib.pyplot as plt
# Load dataset
iris = load_iris()
X = iris.data
y = iris.target
dt = DecisionTreeClassifier(random_state=42)
param_grid = {
    'max_depth': [2, 3, 4, 5, 6, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy']
}
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
grid_search = GridSearchCV(
    estimator=dt,
    param_grid=param_grid,
    cv=kfold,
```

```
    scoring='accuracy',
    n_jobs=-1
)
grid_search.fit(X, y)
print("Best hyperparameters:", grid_search.best_params_)
print("Best cross-validation accuracy:", grid_search.best_score_)
# Train final model using best parameters
best_model = grid_search.best_estimator_
# Generate and display decision tree
plt.figure(figsize=(14, 8))
plot_tree(best_model,
          feature_names=iris.feature_names,
          class_names=iris.target_names,
          filled=True)
plt.title("Best Decision Tree from Grid Search")
plt.show()
```

```
PS C:\Users\annie\OneDrive\Documents\Programming> python "c:\Users\annie\OneDrive\Documents\Programming\Sa
mple 3.py"
Best hyperparameters: {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 2}
Best cross-validation accuracy: 0.9666666666666668
```

Sample 4

```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
# Load dataset
train_df = pd.read_csv('train.csv')

# Separate target and features
y = train_df['SalePrice']
X = train_df.drop(columns=['SalePrice', 'Id'])
# Identify categorical and numerical columns
cat_cols = X.select_dtypes(include=['object']).columns
num_cols = X.select_dtypes(exclude=['object']).columns
# Preprocessing
preprocessor = ColumnTransformer([
    ('cat', OneHotEncoder(handle_unknown='ignore'), cat_cols),
```

```
    ('num', 'passthrough', num_cols)
])
# Decision Tree Regressor
dt_reg = DecisionTreeRegressor(random_state=42)
# Hyperparameter grid
param_grid = {
    'model__max_depth': [3, 5, 7, 10, None],
    'model__min_samples_split': [2, 5, 10, 20],
    'model__min_samples_leaf': [1, 2, 5, 10]
}
# Pipeline
pipeline = Pipeline([
    ('preprocess', preprocessor),
    ('model', dt_reg)
])
# Cross validation setup
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
# Grid Search
grid_search = GridSearchCV(
    estimator=pipeline,
    param_grid=param_grid,
    scoring='neg_root_mean_squared_error',
    cv=kfold,
    n_jobs=-1
)
# Fit grid search
grid_search.fit(X, y)
print("Best hyperparameters:", grid_search.best_params_)
print("Best CV RMSE:", -grid_search.best_score_)
# Train final best model
best_pipeline = grid_search.best_estimator_
# Extract trained decision tree from pipeline
best_tree = best_pipeline.named_steps['model']
# Transform X using preprocessing
X_transformed = best_pipeline.named_steps['preprocess'].transform(X)
```

```
PS C:\Users\annie\OneDrive\Documents\Programming> python "c:\Users\annie\OneDrive\Documents\Programming\Sa
mple 4.py"
Best hyperparameters: {'model__max_depth': 5, 'model__min_samples_leaf': 5, 'model__min_samples_split': 2}
Best CV RMSE: 34063.95173874488
```