Lab Week 7

Session: 2022 – 2026

Submitted by:

Muhammad Saad Akmal 2022-CS-148

Submitted To:

Sir Nauman Shafi

Department of Computer Science

University of Engineering and Technology

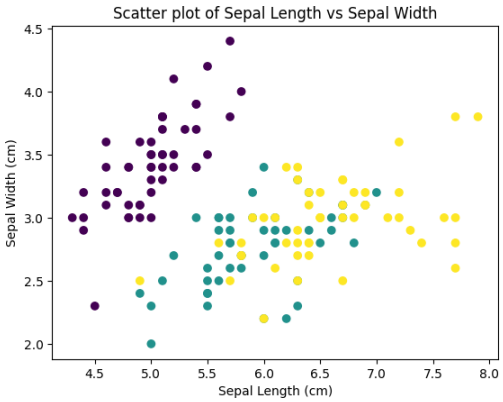
Lahore Pakistan

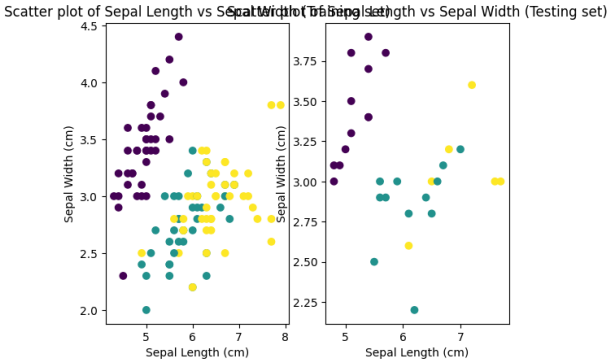
```
''
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import mean_squared_error , r2_score
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

#Q1
iris = datasets.load_iris()
housing = fetch_california_housing()


iris_data = pd.DataFrame(iris.data , columns=iris.feature_names)
iris_target_columns = iris.target_names
iris_target = iris.target
iris_data_null = iris_data.isnull()

plt.scatter(iris_data['sepal length (cm)'], iris_data['sepal width (cm)'], c=iris_target)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter plot of Sepal Length vs Sepal Width')
plt.show()

housing_data = pd.DataFrame(housing.data , columns=housing.feature_names)
hosuing_target_columns = housing.target_names
hosuing_target = housing.target
housing_data_null = housing_data.isnull()


#Q2
iris_scaler = StandardScaler()
iris_scaler.fit(iris_data)
print(iris_scaler.transform(iris_data))


housing_scaler = StandardScaler()
housing_scaler.fit(housing_data)
print(housing_scaler.transform(housing_data))
iris_train_X, iris_test_X , iris_train_Y, iris_test_Y  = train_test_split(iris_data , iris_target , random_state=104,  test_size=0.20,  shuffle=True)

fig, (train, test) = plt.subplots(1, 2)

train.scatter(iris_train_X['sepal length (cm)'], iris_train_X['sepal width (cm)'], c=iris_train_Y)
train.set_xlabel('Sepal Length (cm)')
train.set_ylabel('Sepal Width (cm)')
train.set_title('Scatter plot of Sepal Length vs Sepal Width (Training set)')


test.scatter(iris_test_X['sepal length (cm)'], iris_test_X['sepal width (cm)'], c=iris_test_Y)
test.set_xlabel('Sepal Length (cm)')
test.set_ylabel('Sepal Width (cm)')
test.set_title('Scatter plot of Sepal Length vs Sepal Width (Testing set)')


plt.show()

housing_train_X, housing_test_X ,housing_train_Y, housing_test_Y = train_test_split(housing_data , hosuing_target , random_state=104,  test_size=0.20,  shuffle=True)



#TASK2

#Q3
accuracy_list = []
fig, axes = plt.subplots(2, 2, figsize=(8, 15))
for i in range(1,16):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(iris_train_X , iris_train_Y)
    predicted = neigh.predict(iris_test_X)
    accuracy = metrics.accuracy_score(iris_test_Y, predicted)
    accuracy_list.append(accuracy)
    print(f'For iteration {i} the accuracy is {accuracy}')

axes[0,0].plot([i for i in range(1,16)] , accuracy_list)
axes[0,0].set_xlabel('K')
axes[0,0].set_ylabel('Accuracy')
axes[0,0].set_title('Acuracy of KNN')
```

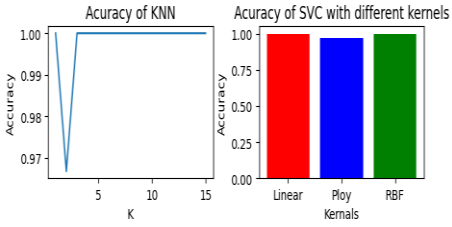
```
#Q4
linear = SVC(kernel='linear').fit(iris_train_X, iris_train_Y)
linear_predict = linear.predict(iris_test_X)
linear_accuracy = metrics.accuracy_score(iris_test_Y , linear_predict)

poly = SVC(kernel='poly').fit(iris_train_X, iris_train_Y)
poly_predict = poly.predict(iris_test_X)
poly_accuracy = metrics.accuracy_score(iris_test_Y , poly_predict)

rbf = SVC(kernel='rbf').fit(iris_train_X, iris_train_Y)
rbf_predict = rbf.predict(iris_test_X)
rbf_accuracy = metrics.accuracy_score(iris_test_Y , rbf_predict)

kernals = ['Linear' , 'Ploy' , 'RBF']
svc_accuracy = [linear_accuracy , poly_accuracy , rbf_accuracy]
axes[0,1].bar(kernals , svc_accuracy , color=['Red' , 'Blue' , 'Green'])
axes[0,1].set_xlabel('Kernals')
axes[0,1].set_ylabel('Accuracy')
axes[0,1].set_title('Acuracy of SVC with different kernels')


#Q5
calssifier_10 = RandomForestClassifier(n_estimators = 10).fit(iris_train_X, iris_train_Y)
predict_10 = calssifier_10.predict(iris_test_X)
acc_10 = metrics.accuracy_score(iris_test_Y , predict_10)
print(f'Accuracy for estimator 10 {acc_10}')

calssifier_50 = RandomForestClassifier(n_estimators = 50).fit(iris_train_X, iris_train_Y)
predict_50 = calssifier_50.predict(iris_test_X)
acc_50 = metrics.accuracy_score(iris_test_Y , predict_50)
print(f'Accuracy for estimator 50 {acc_50}')
#TASK 3

regr = LinearRegression()

regr.fit(housing_train_X, housing_train_Y)
pred = regr.predict(housing_test_X)

mean_square_error = mean_squared_error(housing_test_Y , pred)
r2_Score = r2_score(housing_test_Y,pred)

print(f'Mean square error = {mean_square_error}')
print(f'R2 acore = {r2_Score}')


regressor = DecisionTreeRegressor(random_state = 0)

regressor.fit(housing_train_X, housing_train_Y)
tree_pred = regressor.predict(housing_test_X)

tree_mean_square_error = mean_squared_error(housing_test_Y , tree_pred)
tree_r2_Score = r2_score(housing_test_Y,tree_pred)
print(f'Mean square error = {tree_mean_square_error}')
print(f'R2 acore = {tree_r2_Score}')

indices = range(len(housing_test_Y))


fig, axes = plt.subplots(1, 2)


axes[0].scatter(indices, housing_test_Y, color='blue', alpha=0.6, label='Actual Values')
axes[0].scatter(indices, pred, color='red', alpha=0.6, label='Predicted Values')
axes[0].set_xlabel('Sample Index')
axes[0].set_ylabel('Value')
axes[0].set_title('Actual and Predicted Values (Linear Regression)')
axes[0].legend()
axes[0].grid()


regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(housing_train_X, housing_train_Y)
tree_pred = regressor.predict(housing_test_X)


axes[1].scatter(indices, housing_test_Y, color='blue', alpha=0.6, label='Actual Values')
axes[1].scatter(indices, tree_pred, color='red', alpha=0.6, label='Predicted Values')
axes[1].set_xlabel('Sample Index')
axes[1].set_ylabel('Value')
axes[1].set_title('Actual and Predicted Values (Decision Tree)')
axes[1].legend()
axes[1].grid()


plt.tight_layout()
plt.show()
```

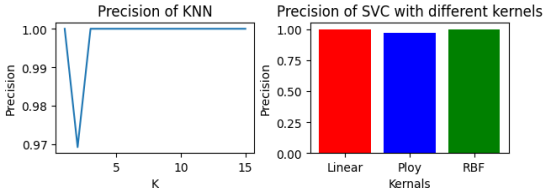
```
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(housing_train_X, housing_train_Y)
tree_pred = regressor.predict(housing_test_X)


axes[1].scatter(indices, housing_test_Y, color='blue', alpha=0.6, label='Actual Values')
axes[1].scatter(indices, tree_pred, color='red', alpha=0.6, label='Predicted Values')
axes[1].set_xlabel('Sample Index')
axes[1].set_ylabel('Value')
axes[1].set_title('Actual and Predicted Values (Decision Tree)')
axes[1].legend()
axes[1].grid()


plt.tight_layout()
plt.show()


model = ['LinearRegression' , 'Decision Tree']
mse = [mean_square_error , tree_mean_square_error]
r2 = [r2_Score , tree_r2_Score]

fig, axes = plt.subplots(1, 2)


axes[0].bar(model, mse, color=['Red', 'Blue'])
axes[0].set_xlabel('Model')
axes[0].set_ylabel('Mean Squared Error (MSE)')
axes[0].set_title('MSE of Models')

# Plot R² Score
axes[1].bar(model, r2, color=['Red', 'Blue'])
axes[1].set_xlabel('Model')
axes[1].set_ylabel('R² Score')
axes[1].set_title('R² Score of Models')

plt.show()
```

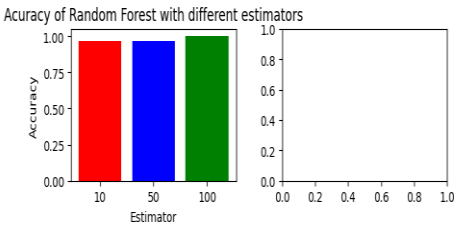
Output:

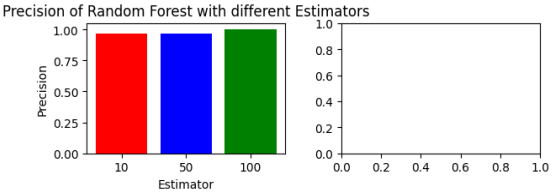


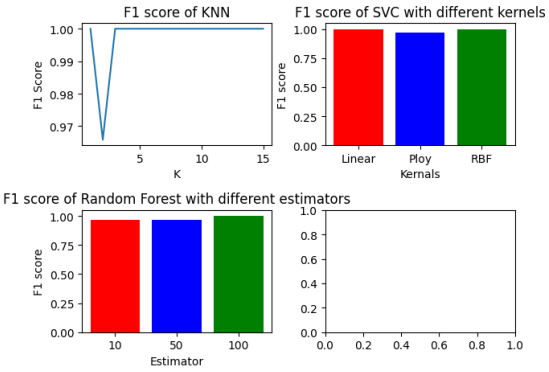


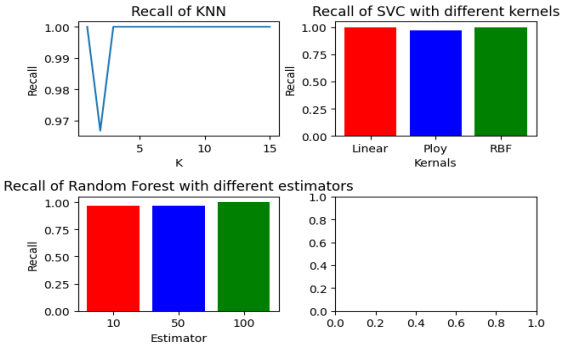


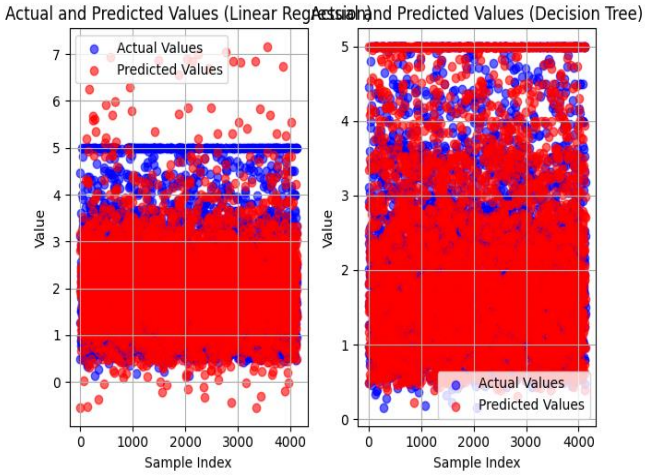


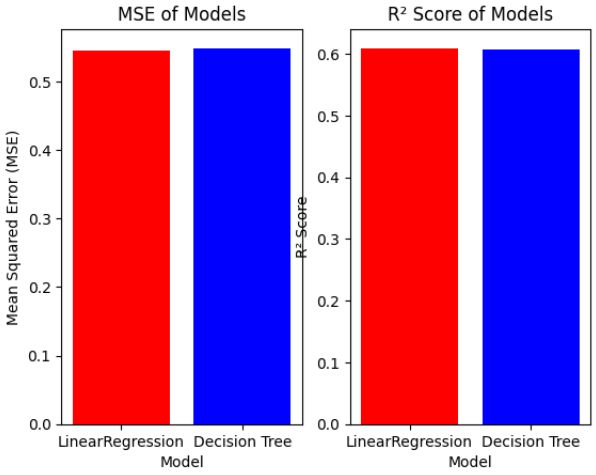





F1 score of KNN
F1 Score
K
F1 score of SVC with different kernels
F1 score
Linear
Ploy
RBF
Kernals
F1 score of Random Forest with different estimators
F1 score
Estimator


Recall of KNN
Recall
K
Recall of SVC with different kernels
Linear
Ploy
RBF
Kernals
Recall of Random Forest with different estimators
Recall
Estimator


Actual and Predicted Values (Linear Regression)
Actual and Predicted Values (Decision Tree)
Actual Values
Predicted Values
Value
Sample Index


MSE of Models
Mean Squared Error (MSE)
LinearRegression
Decision Tree
Model
R² Score of Models
R² Score