# Applying Machine Learning Algorithms


Submitted to:

Mr. Nauman Shafi

Submitted by:

Faisal Ilyas 2022-CS-63

Department of Computer Science
**University of Engineering and Technology**
**Lahore, Pakistan**

## 0.1 Code

```python
# Importing necessary libraries
import pandas as pd
from sklearn.datasets import load_iris, fetch_california_housing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score

# Task 1: Data Preprocessing

# Question 1: Load and Explore the Datasets
# Loading the Iris dataset for classification task
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['Species'] = iris.target

# Loading the California Housing dataset for regression task
california = fetch_california_housing()
california_df = pd.DataFrame(data=california.data, columns=california.feature_names)
california_df['MedianHouseValue'] = california.target

# Exploring the datasets
# Displaying first few rows of Iris dataset
print("Iris Dataset:")
print(iris_df.head())

# Checking for missing values in Iris dataset
print("\nMissing values in Iris dataset:")
print(iris_df.isnull().sum())

# Displaying first few rows of California Housing dataset
print("\nCalifornia Housing Dataset:")
print(california_df.head())

# Checking for missing values in California Housing dataset
print("\nMissing values in California Housing dataset:")
print(california_df.isnull().sum())
```

Figure 1: Code

```python
# Question 2: Feature Scaling and Splitting Data
# Feature scaling for both datasets using StandardScaler
scaler_iris = StandardScaler()
iris_scaled = scaler_iris.fit_transform(iris_df.drop(columns=['Species']))

scaler_california = StandardScaler()
california_scaled = scaler_california.fit_transform(california_df.drop(columns=['MedianHouseValue']))

# Splitting both datasets into training and testing sets (80% training, 20% testing)
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(
    iris_scaled, iris_df['Species'], test_size=0.2, random_state=42
)

X_train_california, X_test_california, y_train_california, y_test_california = train_test_split(
    california_scaled, california_df['MedianHouseValue'], test_size=0.2, random_state=42
)

# Task 2: Classification Algorithms

# Question 3: Apply k-Nearest Neighbors (k-NN) for Classification
# Training k-NN model for different values of k and evaluating accuracy
for k in range(1, 4):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_iris, y_train_iris)
    y_pred_iris = knn.predict(X_test_iris)
    accuracy = accuracy_score(y_test_iris, y_pred_iris)
    print(f'k = {k}, k-NN Accuracy = {accuracy}')

# Question 4: Apply Support Vector Machine (SVM) for Classification
# Training SVM model with different kernels and evaluating accuracy
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for kernel in kernels:
    svm = SVC(kernel=kernel)
    svm.fit(X_train_iris, y_train_iris)
    y_pred_iris = svm.predict(X_test_iris)
    accuracy = accuracy_score(y_test_iris, y_pred_iris)
    print(f'SVM (kernel={kernel}) Accuracy = {accuracy}')
```

Figure 2: Code

```python
# Question 5: Apply Random Forest Classifier
# Training Random Forest Classifier with 10 estimators and evaluating accuracy
rf = RandomForestClassifier(n_estimators=10)
rf.fit(X_train_iris, y_train_iris)
y_pred_iris = rf.predict(X_test_iris)
accuracy = accuracy_score(y_test_iris, y_pred_iris)
print(f'Random Forest (10 estimators) Accuracy = {accuracy}')

# Task 3: Regression Algorithms

# Question 6: Apply Linear Regression for Regression
# Training Linear Regression model on California Housing dataset and evaluating performance
lr = LinearRegression()
lr.fit(X_train_california, y_train_california)
y_pred_california = lr.predict(X_test_california)
mse_lr = mean_squared_error(y_test_california, y_pred_california)
r2_lr = r2_score(y_test_california, y_pred_california)
print(f'Linear Regression MSE = {mse_lr}, R² = {r2_lr}')

# Question 7: Apply Decision Tree Regression
# Training Decision Tree Regressor and comparing with Linear Regression using MSE and R² score
dt = DecisionTreeRegressor()
dt.fit(X_train_california, y_train_california)
y_pred_california = dt.predict(X_test_california)
mse_dt = mean_squared_error(y_test_california, y_pred_california)
r2_dt = r2_score(y_test_california, y_pred_california)
print(f'Decision Tree MSE = {mse_dt}, R² = {r2_dt}')

# Task 4: Model Evaluation and Comparison

# Question 8: Evaluate Classification Models Using Classification Metrics
# Comparing k-NN, SVM, and Random Forest models based on accuracy
print(f'Final k-NN Accuracy: {accuracy_score(y_test_iris, knn.predict(X_test_iris))}')
print(f'Final SVM Accuracy: {accuracy_score(y_test_iris, svm.predict(X_test_iris))}')
print(f'Final Random Forest Accuracy: {accuracy_score(y_test_iris, rf.predict(X_test_iris))}')

# Question 9: Evaluate Regression Models Using Regression Metrics
# Comparing Linear Regression and Decision Tree models using MSE and R² score
print(f'Final Linear Regression MSE: {mse_lr}, R²: {r2_lr}')
print(f'Final Decision Tree MSE: {mse_dt}, R²: {r2_dt}')
```

Figure 3: Code

```python
# Task 5: Conclusion

# Question 10: Compare and Summarize the Findings
# Summary of classification models:
# - k-NN achieved perfect accuracy for k=2 and k=3.
# - SVM with the RBF kernel also reached perfect accuracy, while the linear and polynomial kernels performed slightly lower.
# - Random Forest classifier achieved perfect accuracy, making it one of the most reliable models.
#
# Summary of regression models:
# - Linear Regression resulted in a moderate prediction error with an MSE of 0.556.
# - Decision Tree Regressor slightly outperformed Linear Regression with a lower MSE of 0.505.
```

Figure 4: Code

# 1 Output

```
Iris Dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  Species
0                5.1               3.5                1.4               0.2        0
1                4.9               3.0                1.4               0.2        0
2                4.7               3.2                1.3               0.2        0
3                4.6               3.1                1.5               0.2        0
4                5.0               3.6                1.4               0.2        0

Missing values in Iris dataset:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
Species              0
dtype: int64

California Housing Dataset:
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  Longitude  MedianHouseValue
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88    -122.23             4.526
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86    -122.22             3.585
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85    -122.24             3.521
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85    -122.25             3.413
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85    -122.25             3.422

Missing values in California Housing dataset:
MedInc             0
HouseAge           0
AveRooms           0
AveBedrms          0
Population         0
AveOccup           0
Latitude           0
Longitude          0
MedianHouseValue   0
dtype: int64
k = 1, k-NN Accuracy = 0.9666666666666667
k = 2, k-NN Accuracy = 1.0
k = 3, k-NN Accuracy = 1.0
SVM (kernel=linear) Accuracy = 0.9666666666666667
SVM (kernel=poly) Accuracy = 0.9666666666666667
SVM (kernel=rbf) Accuracy = 1.0
SVM (kernel=sigmoid) Accuracy = 0.9
Random Forest (10 estimators) Accuracy = 1.0
Linear Regression MSE = 0.5558915986952444, R² = 0.5757877060324508
Decision Tree MSE = 0.4983876455466812, R² = 0.6196701534999252
Final k-NN Accuracy: 1.0
Final SVM Accuracy: 0.9
Final Random Forest Accuracy: 1.0
Final Linear Regression MSE: 0.5558915986952444, R²: 0.5757877060324508
Final Decision Tree MSE: 0.4983876455466812, R²: 0.6196701534999252
Live Share
```

Figure 5: Code