PRACTICAL TECHNICAL ASSESMENT

ARCHANA G S

NSTI CALICUT

Activity

1. Load the dataset and apply necessary preprocessing steps.

2. Perform exploratory data analysis (EDA) to understand the dataset.

3. Implement classification models and evaluate them using a confusion matrix and

cross-validation.

4. Implement regression models and evaluate them using R-squared, MSE, and crossvalidation.

5. Visualize the confusion matrix for at least one classification model.

6. Report and interpret the results of each model.

Requirements

- Personal computer/laptop
- Google Collab
- Dataset (data.csv)

Procedure

Data Preprocessing

- ✓ Load the dataset using pd.read_csv('data.csv')
- ✓ Handle missing values.
- ✓ Encode categorical variables.
- ✓ Scale/normalize the features.

```
# Load the dataset.
import pandas as pd
data = pd.read_csv('data.csv')
print(data)
```

[3] ✓ 0.0s

```
     feature1  feature2  feature3  feature4  target
0         5.1       3.5       1.4       0.2  Class1
1         4.9       3.0       1.4       0.2  Class1
2         4.7       3.2       1.3       0.2  Class1
3         4.6       3.1       1.5       0.2  Class1
4         5.0       3.6       1.4       0.2  Class1
..        ...       ...       ...       ...     ...
144       6.7       3.0       5.2       2.3  Class3
145       6.3       2.5       5.0       1.9  Class3
146       6.5       3.0       5.2       2.0  Class3
147       6.2       3.4       5.4       2.3  Class3
148       5.9       3.0       5.1       1.8  Class3

[149 rows x 5 columns]
```

```
#Handle missing values.
data.dropna(inplace=True)
print(data)
```

[17]

```
     feature1  feature2  feature3  feature4  target
0         5.1       3.5       1.4       0.2  Class1
1         4.9       3.0       1.4       0.2  Class1
2         4.7       3.2       1.3       0.2  Class1
3         4.6       3.1       1.5       0.2  Class1
4         5.0       3.6       1.4       0.2  Class1
..        ...       ...       ...       ...     ...
144       6.7       3.0       5.2       2.3  Class3
145       6.3       2.5       5.0       1.9  Class3
146       6.5       3.0       5.2       2.0  Class3
147       6.2       3.4       5.4       2.3  Class3
148       5.9       3.0       5.1       1.8  Class3

[149 rows x 5 columns]
```

```
# Encode categorical variables.
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['target'] = le.fit_transform(data['target'])
print(data)
```

[18]

```
     feature1  feature2  feature3  feature4  target
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
..        ...       ...       ...       ...     ...
144       6.7       3.0       5.2       2.3       2
145       6.3       2.5       5.0       1.9       2
146       6.5       3.0       5.2       2.0       2
147       6.2       3.4       5.4       2.3       2
148       5.9       3.0       5.1       1.8       2

[149 rows x 5 columns]
```

```
# Scale/normalize the features.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data[['feature1', 'feature2', 'feature3', 'feature4']] = scaler.fit_transform(
print(data)
```

[19]

```
     feature1  feature2  feature3  feature4  target
0   -0.911029  1.023017 -1.347869 -1.325063       0
1   -1.153861 -0.126527 -1.347869 -1.325063       0
2   -1.396694  0.333290 -1.404730 -1.325063       0
3   -1.518110  0.103382 -1.291008 -1.325063       0
4   -1.032445  1.252925 -1.347869 -1.325063       0
..        ...       ...       ...       ...     ...
144  1.031630 -0.126527  0.812843  1.442709       2
145  0.545966 -1.276070  0.699121  0.915514       2
146  0.788798 -0.126527  0.812843  1.047313       2
147  0.424549  0.793108  0.926565  1.442709       2
148  0.060301 -0.126527  0.755982  0.783716       2

[149 rows x 5 columns]
```

```python
#Exploratory Data Analysis (EDA)
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('data.csv')
# Statistical summaries
print(data.describe())
```

[4] ✓ 0.1s

```
            feature1    feature2    feature3    feature4
count  149.000000  149.000000  149.000000  149.000000
mean     5.850336    3.055034    3.770470    1.205369
std      0.826391    0.436422    1.764611    0.761292
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
```

Exploratory Data Analysis (EDA)

- Provide statistical summaries of the dataset.
- Visualize the data distribution and relationships between features using plots.

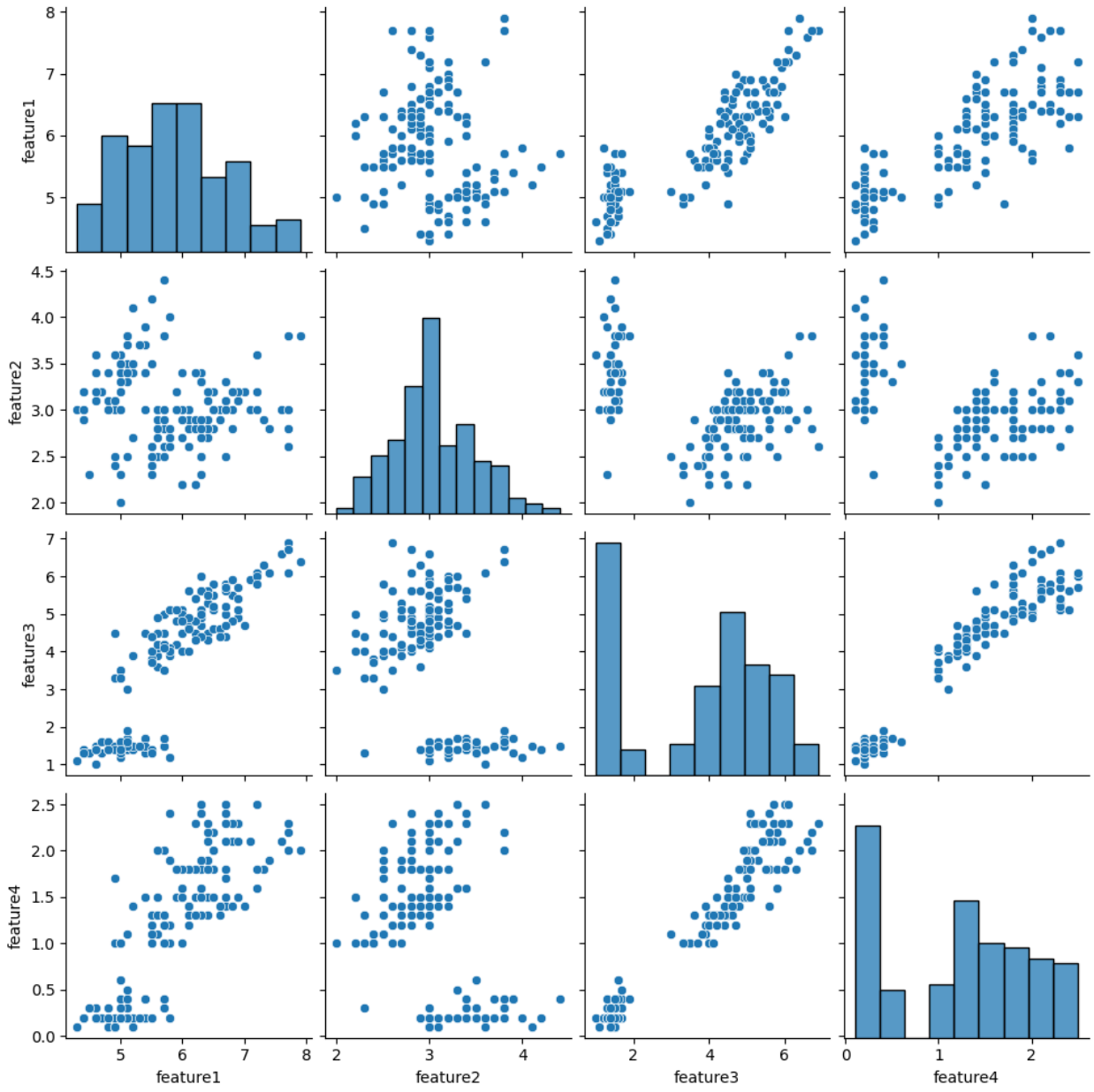
```
#Exploratory Data Analysis (EDA)
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('data.csv')
# Statistical summaries
print(data.describe())
```

[5] ✓ 0.0s

```
...          feature1    feature2    feature3    feature4
count  149.000000  149.000000  149.000000  149.000000
mean     5.850336    3.055034    3.770470    1.205369
std      0.826391    0.436422    1.764611    0.761292
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
```

```
# Visualize the data distribution and relationships
plt.figure(figsize=(10, 4))
sns.pairplot(data )
plt.show()
```

[6] ✓ 3.2s


feature1
feature2
feature3
feature4
feature1
feature2
feature3
feature4

Classification

- Apply Logistic Regression, Decision Tree, and Random Forest classifiers.
- Use a confusion matrix to evaluate the performance of each classifier.
- Perform cross-validation to assess the model stability.

```
import pandas as pd
from sklearn.model_selection import train_test_split
df = pd.read_csv('data.csv')
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

[7] ✓ 0.5s

```
# Split the dataset into training and testing sets
X = df[['feature1', 'feature2', 'feature3', 'feature4']]
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
# Confusion Matrix
conf_matrix_lr = confusion_matrix(y_test, y_pred_lr)
print("Logistic Regression:")
print("Confusion Matrix:\n", conf_matrix_lr)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
```

[8] ✓ 0.0s

```
Logistic Regression:
Confusion Matrix:
 [[10  0  0]
 [ 0  6  3]
 [ 0  0 11]]
Accuracy: 0.9
```

```
# Precision, Recall, F1 Score for multiclass
average_method = 'weighted'  # or 'micro', 'macro', depending on your requirement

print("Precision:", precision_score(y_test, y_pred_lr, average=average_method))
print("Recall:", recall_score(y_test, y_pred_lr, average=average_method))
print("F1 Score:", f1_score(y_test, y_pred_lr, average=average_method))
```

[9] ✓ 0.0s

```
Precision: 0.9214285714285714
Recall: 0.9
F1 Score: 0.896
```

```
# Decision Tree Classifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)

# Confusion Matrix
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
print("\nDecision Tree Classifier:")
print("Confusion Matrix:\n", conf_matrix_dt)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
```

[10] ✓ 0.0s

```
Decision Tree Classifier:
Confusion Matrix:
 [[10  0  0]
 [ 0  6  3]
 [ 0  0 11]]
Accuracy: 0.9
```

```
# Precision, Recall, F1 Score for multiclass
average_method = 'weighted'  # or 'micro', 'macro', depending on your requirement

print("Precision:", precision_score(y_test, y_pred_dt, average=average_method))
print("Recall:", recall_score(y_test, y_pred_dt, average=average_method))
print("F1 Score:", f1_score(y_test, y_pred_dt, average=average_method))
```

[11] ✓ 0.0s

```
Precision: 0.9214285714285714
Recall: 0.9
F1 Score: 0.896
```

```
# Random Forest Classifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Confusion Matrix
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
print("\nRandom Forest Classifier:")
print("Confusion Matrix:\n", conf_matrix_rf)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
```

[12] ✓ 0.1s

```
Random Forest Classifier:
Confusion Matrix:
 [[10  0  0]
 [ 0  6  3]
 [ 0  0 11]]
Accuracy: 0.9
```

Regression

- Apply Linear Regression and Decision Tree Regressor.
- Evaluate the models using R-squared and Mean Squared Error (MSE).
- Perform cross-validation to assess the model stability.
-

```python
# Regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score, mean_squared_error


# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
r2_lr = r2_score(y_test, y_pred_lr)
mse_lr = mean_squared_error(y_test, y_pred_lr)
print("\nLinear Regression:")
print("R-squared:", r2_lr)
print("Mean Squared Error:", mse_lr)

# Decision Tree Regressor
dtr = DecisionTreeRegressor()
dtr.fit(X_train, y_train)
y_pred_dtr = dtr.predict(X_test)
r2_dtr = r2_score(y_test, y_pred_dtr)
mse_dtr = mean_squared_error(y_test, y_pred_dtr)
print("\nDecision Tree Regressor:")
print("R-squared:", r2_dtr)
print("Mean Squared Error:", mse_dtr)
```

```
Linear Regression:
R-squared: 0.9165749856447737
Mean Squared Error: 0.0583048155882637

Decision Tree Regressor:
R-squared: 0.8569157392686804
Mean Squared Error: 0.1
```

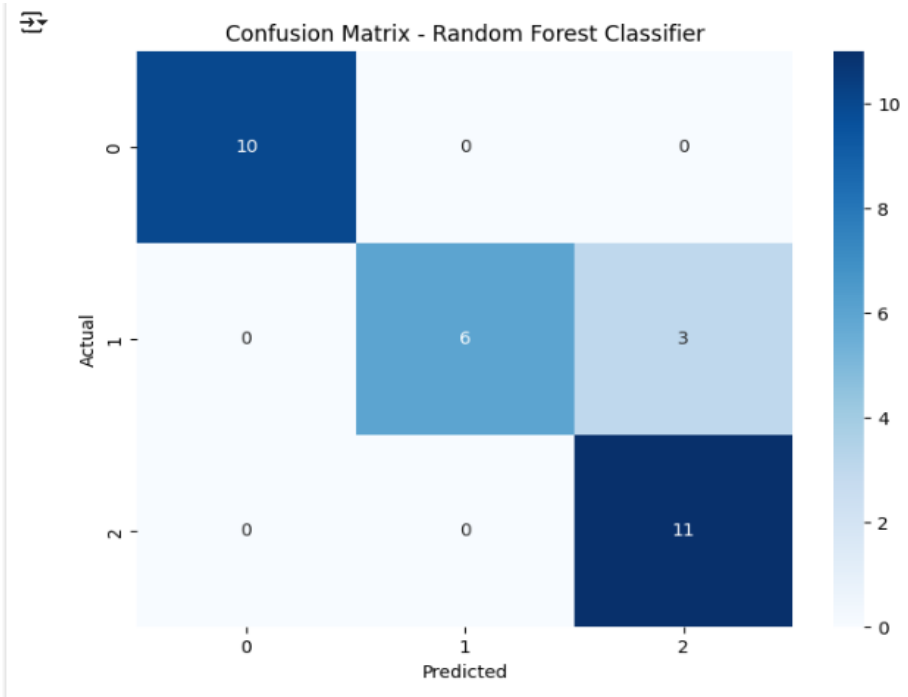
Confusion Matrix

For classification tasks, plot the confusion matrix and compute the following metrics:

- Accuracy
- Precision
- Recall
- F1 Score

```
# Confusion Matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the confusion matrix for the Random Forest Classifier
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_rf, annot=True, cmap='Blues', fmt='g')
plt.title('Confusion Matrix - Random Forest Classifier')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



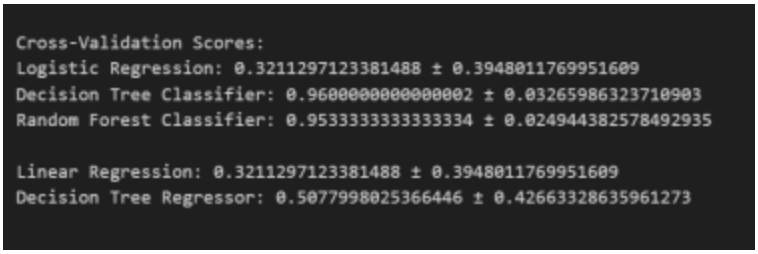
Cross-Validation

- Implement k-fold cross-validation for both classification and regression models.
- Report the mean and standard deviation of the cross-validation scores.

```
# 6. Cross-Validation
from sklearn.model_selection import cross_val_score

# Cross-validation for Classification Models
print("\nCross-Validation Scores:")
print("Logistic Regression:", cross_val_score(lr, X, y, cv=5).mean(), "±", cross_val_score(lr, X, y, cv=5).std())
print("Decision Tree Classifier:", cross_val_score(dt, X, y, cv=5).mean(), "±", cross_val_score(dt, X, y, cv=5).std())
print("Random Forest Classifier:", cross_val_score(rf, X, y, cv=5).mean(), "±", cross_val_score(rf, X, y, cv=5).std())

# Cross-validation for Regression Models
print("\nLinear Regression:", cross_val_score(lr, X, y, cv=5, scoring='r2').mean(), "±", cross_val_score(lr, X, y, cv=5, scoring='r2').std())
print("Decision Tree Regressor:", cross_val_score(dtr, X, y, cv=5, scoring='r2').mean(), "±", cross_val_score(dtr, X, y, cv=5, scoring='r2').std())
```



```
Cross-Validation Scores:
Logistic Regression: 0.3211297123381488 ± 0.3948011769951609
Decision Tree Classifier: 0.9600000000000002 ± 0.03265986323710903
Random Forest Classifier: 0.9533333333333334 ± 0.024944382578492935

Linear Regression: 0.3211297123381488 ± 0.3948011769951609
Decision Tree Regressor: 0.5077998025366446 ± 0.42663328635961273
```

Conclusion

This documentation outlines the process of loading and preprocessing a dataset, performing exploratory data analysis (EDA), implementing classification and regression models, evaluating the models using various metrics, and visualizing results. The dataset used is assumed to have numerical and categorical features, with a target variable for prediction.