ML DL Mcq

1. A

2. A

3. B

4. C

5. B

6. A

Question 1: Binary Classification with scikit-learn

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [ ]:
```
data = pd.read_csv('data.csv')

# Separate features (age and income) and target labels
X = data[['age', 'income']]
y = data['purchase']
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:
```
# Create and train the model
model = LogisticRegression()
model.fit(X_train, y_train)
```

In [ ]:
```
y_pred = model.predict(X_test)
```

In [ ]:
```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)

# Classification Report
class_report = classification_report(y_test, y_pred)
print('Classification Report:')
print(class_report)
```

Question 2: Multiclass Classification with Dummy Data

In [ ]:
```
# Sample dataset
data = {
    'weight': [140, 130, 150, 160, 155, 175, 120, 110, 145, 165],
    'color (1=Red , 2=Yellow , 3=Orange)': [1,2,2,1,3,2,3,2,1,3],
    'fruit': ['Apple', 'Banana', 'Apple', 'Orange', 'Banana', 'Orange', 'Apple', 'Banana', 'Apple', 'Orange']
}

df = pd.DataFrame(data)
```

In [ ]:
```
X = df[['weight', 'color_encoded']]
y = df['fruit']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:
```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

In [ ]:
```
y_pred = clf.predict(X_test)
```

In [ ]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')


classification_rep = classification_report(y_test, y_pred)
print(f'Classification Report:{classification_rep}')

# Generate a confusion matrix
confusion_mtx = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:{confusion_mtx}')
```

Question 3: Clustering with scikit-learn

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [ ]:
```
np.random.seed(0)
n_samples = 200
X = np.random.rand(n_samples, 2)
```

In [ ]:
```
n_clusters = 3
```

In [ ]:
```
kmeans = KMeans(n_clusters=n_clusters)
kmeans.fit(X)
```

In [ ]:
```
cluster_labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

In [ ]:
```
plt.figure(figsize=(8, 6))
colors = ['r', 'g', 'b']

for i in range(n_clusters):
    plt.scatter(X[cluster_labels == i, 0], X[cluster_labels == i, 1], s=50, c=colors[i], label=f'Cluster {i+1}')

plt.scatter(centroids[:, 0], centroids[:, 1], s=100, c='k', marker='X', label='Centroids')
plt.xlabel('Age')
plt.ylabel('Spending Score')
plt.legend()
plt.title('K-Means Clustering')
plt.show()
```

Question 4: Regression with scikit-learn

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [ ]:
```
data = pd.read_csv("data")


X = data[['bedrooms', 'square_footage']]
y = data['price']
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

In [ ]:
```
y_pred = model.predict(X_test)
```

In [ ]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

Pandas and numpy mcqs

1. A

2. B

3. A

4. B

5. A

6. A

7. A

8. A

9. B

10. B

Programming questions

Question 1

In [ ]:
```
import pandas as pd


data = {
    'name': ['Alice', 'Bob', 'Charlie'],
    'age': [25, 30, 22]
}

df = pd.DataFrame(data)


def print_age_of_person(name):
    person = df[df['name'] == name]
    if not person.empty:
        age = person['age'].values[0]
        print(f"The age of {name} is {age}")
    else:
        print(f"No record found for {name}")

def calculate_average_age():
    avg_age = df['age'].mean()
    print(f"The average age of all individuals is {avg_age:.2f}")

print_age_of_person('Alice')
calculate_average_age()
```

Question 3

In [ ]:
```
random_values = np.random.rand(20)


print("Random Array:")
print(random_values)


def calculate_mean(arr):
    return np.mean(arr)


def count_values_above_threshold(arr, threshold):
    count = np.sum(arr > threshold)
    return count


mean = calculate_mean(random_values)
count_above_0_5 = count_values_above_threshold(random_values, 0.5)

print(f"Mean of the array: {mean:.2f}")
print(f"Number of values greater than 0.5: {count_above_0_5}")
```

Question 6

In [ ]:
```
random_values = np.random.rand(20)


print("Random Array:")
print(random_values)


def calculate_median(arr):
    return np.median(arr)


def count_values_above_threshold(arr, threshold):
    count = np.sum(arr > threshold)
    return count


median = calculate_median(random_values)
```

```
count_above_0_3 = count_values_above_threshold(random_values, 0.3)

print(f"Median of the array: {median:.2f}")
print(f"Number of values greater than 0.3: {count_above_0_3}")
```