3/30/2025

Nagarajan

SIRPI ERIP BATCH 2

Machine language Test

***Name : Nagarajan C***

***Section 1: Multiple Choice Questions***

*1) a) pd.read\_csv()*

*2) b) NumPy*

*3) a) df.head()*

*4) c) It consists of rows and columns like a table*

*5) b) K-Means Clustering*

*6) c) To evaluate model performance on unseen data*

*7) c) Mean Squared Error (MSE)*

*8) c) Matplotlib*

*9) a) lambda x: x \* 2*

*10) c) It brings numerical features to a common scale*

*11) b) Histogram*

*12) c) Box Plot*

*13) a) Univariate deals with one variable, bivariate deals with two variables*

*14) d) All of the above*

*15) b) Scatter Plot*

*16) d) All of the above*

*17) a) df.isnull()*

*18) b) Bar Chart*

*19) b) Min-Max Scaling*

*20) a) A correlation coefficient of 0 means no relationship exists between variables*

**Section 2:**

**1)Standardization Explanation:**  
The code standardizes the data using StandardScaler, transforming each feature to have a mean of 0 and a standard deviation of 1. This is important in Machine Learning to ensure features contribute equally to model performance.

**2)**

import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],

'Age': [25, 30, 35, None],

'Salary': [50000, 60000, None, 80000]}

df = pd.DataFrame(data)

df['Age'].fillna(df['Age'].mean(), inplace=True)

df.dropna(subset=['Salary'], inplace=True)

print(df)

**3)**

**import pandas as pd**

**df = pd.DataFrame({'A': [10, 20, 30, 40], 'B': [5, 15, 25, 35]})**

**df['C'] = df['A'].apply(lambda x: x \* 2) + df['B'].map(lambda x: x - 2)**

**print(df)**

df['C'] = df['A'].apply(lambda x: x \* 2) + df['B'].map(lambda x: x - 2)

in this line of code, the lambda function multiplies each value in column 'A' by 2,then subtracts 2 from each value in column 'B' and add df[‘A] and df[‘B].

create the new column 'C' by adding the results from the apply() and map() functions for each row.

**Predicted output:**

A B C

0 10 5 23

1 20 15 53

2 30 25 83

3 40 35 113

**The role of apply () and map() in the above code :**

The apply () function is used to apply a function to each element of the specified column.

The map () function is used to map a function to each element in a Series, meaning it operates element-wise on the data in the Series.

4) **correlation heatmap**

A correlation heatmap is a graphical representation of the correlation matrix, where each cell represents the correlation between two variables.

It helps to find the relationship between variables whether positive or negative correlation

It uses colors to show the strength and direction of the correlation.

**Positive correlation:**

When two variables increase together .

**Negative correlation:**

one variable increases when the other one decreases .

**No correlation:**

When there is no linear relationship.

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

data = {'A': [1, 2, 3, 4, 5], 'B': [5, 4, 3, 2, 1], 'C': [2, 4, 6, 8, 10]}

df = pd.DataFrame(data)

corr\_matrix = df.corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)

plt.show()

**5) The difference between a histogram and a bar chart.**

**Histogram:**

Histogram is Used to show the distribution of continuous numerical data.

In Histogram the Data is grouped into bins.

The bars in the Histogram represent the frequency or count of data points in each interval.

**Bar Chart:**

Bar chart are used to compare categorical data.

Each bar represents a category, and its height represents the value or frequency of that category.

import matplotlib.pyplot as plt

data = [5, 10, 15, 20, 25]

# Create Histogram

plt.subplot(1, 2, 1)

plt.hist(data, bins=3, color='skyblue')

plt.title('Histogram')

# Data for bar chart (categorical data)

categories = ['A', 'B', 'C']

values = [5, 10, 15]

# Create Bar Chart

plt.subplot(1, 2, 2)

plt.bar(categories, values, color='orange')

plt.title('Bar Chart')

plt.tight\_layout()

plt.show()

1. **A company wants to analyse the relationship between advertising budget and sales revenue.**

**Which type of plot should be used & Why?**

In this case we can use scatter plots because It helps to visually see if there is any correlation between the two variables.

It shows how changes in the advertising budget relate to changes in sales revenue.

Helps identify patterns (positive or negative correlation), clusters, or outliers in the data.

import matplotlib.pyplot as plt

# Sample data: Advertising budget and sales revenue

advertising\_budget = [1000, 2000, 3000, 4000, 5000]

sales\_revenue = [10, 20, 30, 40, 50]

# Create Scatter Plot

plt.scatter(advertising\_budget, sales\_revenue, color='blue')

plt.title('Advertising Budget vs Sales Revenue')

plt.xlabel('Advertising Budget')

plt.ylabel('Sales Revenue')

# Show the plot

plt.show()

**7) The difference between Label Encoding and One-Hot Encoding.**

**Label Encoding:**

Converts each category into a unique integer.

It assigns each unique category a number .

It is useful when the categorical feature has an ordinal relationship ,like one category has a higher or lower ranking than the other.

**One-Hot Encoding:**

Converts each category into a binary vector ,a series of 0s and 1s.

Each category is represented as a separate column where 1 indicates the presence of that category and 0 indicates absence.

It is useful when the categorical feature has no nominal data.

import pandas as pd

from sklearn.preprocessing import OneHotEncoder

df = pd.DataFrame({'Color': ['Red', 'Blue', 'Green', 'Blue', 'Red']})

encoder = OneHotEncoder(sparse=False)

encoded\_colors = encoder.fit\_transform(df[['Color']])

encoded\_df = pd.DataFrame(encoded\_colors, columns=encoder.categories\_[0])

df\_encoded = pd.concat([df, encoded\_df], axis=1)

print(df\_encoded)

We can use Label Encoding when the categorical variable is ordinal, like,Low, Medium, High.

We can use One-Hot Encoding when the categorical variable is nominal like,'Red', 'Blue', 'Green'.

**8)**

import pandas as pd

data = {'Category': ['A', 'B', 'A', 'B', 'C', 'A'],

'Sales': [100, 200, 150, 250, 300, 120]}

df = pd.DataFrame(data)

meanofsales = df.groupby('Category')['Sales'].mean()

print(meanofsales)