Module 1 syllabus

Need for data science – Benefits and uses – Facets of data – Data Science process – Setting the research goal – Retrieving data – Cleansing, integrating, and transforming data – Exploratory data analysis – Build the models – Presenting and building applications

1.Need for Data Science

Explanation: Data science is essential to uncover meaningful insights from data, enabling informed decision-making. In the Titanic dataset, for example, we can analyze survival trends to understand patterns and make predictions.

Benefits of Data Science:

Predict outcomes (e.g., survival probability). Handle large datasets efficiently. Optimize business processes.

```
# Load Titanic dataset
import seaborn as sns
import pandas as pd

df = sns.load_dataset('titanic')

# Analyze survival rates based on gender
survival_by_gender = df.groupby('sex')['survived'].mean()
print("Survival Rates by Gender:\n", survival_by_gender)

# Analyze survival rates based on passenger class
survival_by_class = df.groupby('class')['survived'].mean()
print("\nSurvival Rates by Class:\n", survival_by_class)
```

```
Survival Rates by Gender:
 sex
female    0.742038
male      0.188908
Name: survived, dtype: float64

Survival Rates by Class:
 class
First     0.629630
Second    0.472826
Third     0.242363
Name: survived, dtype: float64
<ipython-input-2-a4a95cd4ebe8>:12: FutureWarning: The default of observed=False is deprecated and will be changed to True in a futur
  survival_by_class = df.groupby('class')['survived'].mean()
```

Output Explanation:

This code calculates survival rates by gender and class. For example, females may have a higher survival rate, and first-class passengers often survive more than third-class passengers.

2.Facets of Data

Explanation:

Structured Data: Organized data in rows and columns (e.g., Age, Fare).

Categorical Data: Data with specific categories (e.g., Sex, Embarked).

Missing Data: Data that is incomplete and requires preprocessing.

```
# Check the structure of the dataset
print("Data Types:\n", df.dtypes)

# Check for missing values
missing_values = df.isnull().sum()
print("\nMissing Values:\n", missing_values)

# Display data categories
categorical_columns = df.select_dtypes(include='category').columns
print("\nCategorical Columns:\n", categorical_columns)
```

```
Data Types:
 survived         int64
pclass           int64
sex             object
age            float64
sibsp            int64
parch            int64
fare           float64
embarked        object
```

```
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object

Missing Values:
 survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64

Categorical Columns:
 Index(['class', 'deck'], dtype='object')
```

Output Explanation:

Data types and missing values in the dataset are identified.

Example: Age has missing values, and Sex is categorical.

3.Data Science Process

The process involves:

Data Understanding: Loading and exploring data.

Data Preparation: Cleaning and preprocessing.

Model Building: Using machine learning to build predictive models.

Evaluation: Testing model performance.

Deployment: Using models for real-world applications.

```
# Step 1: Load and explore
print("Preview of Data:\n", df.head())

# Step 2: Data preparation
df_cleaned = df.dropna()  # Drop rows with missing values
print("\nAfter Cleaning:\n", df_cleaned.head())

# Step 3: Build a model (Logistic Regression)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Select features and target
X = df_cleaned[['age', 'fare']]
y = df_cleaned['survived']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a model
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 4: Evaluate
accuracy = model.score(X_test, y_test)
print("\nModel Accuracy:", accuracy)
```

```
Preview of Data:
    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

      who  adult_male deck  embark_town alive  alone
```

```
    0    man        True  NaN  Southampton    no  False
    1  woman       False    C    Cherbourg   yes  False
    2  woman       False  NaN  Southampton   yes   True
    3  woman       False    C  Southampton   yes  False
    4    man        True  NaN  Southampton    no   True

    After Cleaning:
        survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
    1          1       1  female  38.0      1      0  71.2833        C  First
    3          1       1  female  35.0      1      0  53.1000        S  First
    6          0       1    male  54.0      0      0  51.8625        S  First
    10         1       3  female   4.0      1      1  16.7000        S  Third
    11         1       1  female  58.0      0      0  26.5500        S  First

          who  adult_male deck  embark_town alive  alone
    1   woman       False    C    Cherbourg   yes  False
    3   woman       False    C  Southampton   yes  False
    6     man        True    E  Southampton    no   True
    10  child       False    G  Southampton   yes  False
    11  woman       False    C  Southampton   yes   True

    Model Accuracy: 0.7297297297297297
```

Output Explanation:

Data is cleaned, and a basic logistic regression model predicts survival. Model accuracy indicates how well it predicts on unseen data.

4.Setting the Research Goal

Explanation: A clear research goal guides analysis.

Example Goal: Predict survival based on age, gender, and passenger class.

```
# Define the research goal
print("Research Goal: Predict survival using Age, Gender, and Class.")
```

```
Research Goal: Predict survival using Age, Gender, and Class.
```

5.Retrieving Data

Explanation:

Retrieve data from a library or external source like Seaborn or Kaggle.

```
# Load dataset
df = sns.load_dataset('titanic')
print("Data Retrieved Successfully!")
print(df.head())
```

```
Data Retrieved Successfully!
       survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
    0         0       3    male  22.0      1      0   7.2500        S  Third
    1         1       1  female  38.0      1      0  71.2833        C  First
    2         1       3  female  26.0      0      0   7.9250        S  Third
    3         1       1  female  35.0      1      0  53.1000        S  First
    4         0       3    male  35.0      0      0   8.0500        S  Third

         who  adult_male deck  embark_town alive  alone
    0    man        True  NaN  Southampton    no  False
    1  woman       False    C    Cherbourg   yes  False
    2  woman       False  NaN  Southampton   yes   True
    3  woman       False    C  Southampton   yes  False
    4    man        True  NaN  Southampton    no   True
```

6.Cleansing, Integrating, and Transforming Data

Explanation: Prepare data for analysis by handling missing values, encoding categorical variables, and scaling.

```
# Clean missing values
df_cleaned = df.dropna(subset=['age', 'embarked'])

# Encode categorical variables
df_cleaned['sex'] = df_cleaned['sex'].map({'male': 0, 'female': 1})

# Scaling numerical features (e.g., age, fare)
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_cleaned[['age', 'fare']] = scaler.fit_transform(df_cleaned[['age', 'fare']])
print("Transformed Data:\n", df_cleaned.head())
```

```
Transformed Data:
   survived  pclass  sex       age  sibsp  parch      fare embarked  class  \
0         0       3    0  0.271174      1      0  0.014151        S  Third
1         1       1    1  0.472229      1      0  0.139136        C  First
2         1       3    1  0.321438      0      0  0.015469        S  Third
3         1       1    1  0.434531      1      0  0.103644        S  First
4         0       3    0  0.434531      0      0  0.015713        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
<ipython-input-7-5642786e7619>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df_cleaned['sex'] = df_cleaned['sex'].map({'male': 0, 'female': 1})
<ipython-input-7-5642786e7619>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df_cleaned[['age', 'fare']] = scaler.fit_transform(df_cleaned[['age', 'fare']])
```

Output Explanation:

Missing values are handled, categorical variables encoded, and numerical features scaled.

7.Exploratory Data Analysis (EDA)

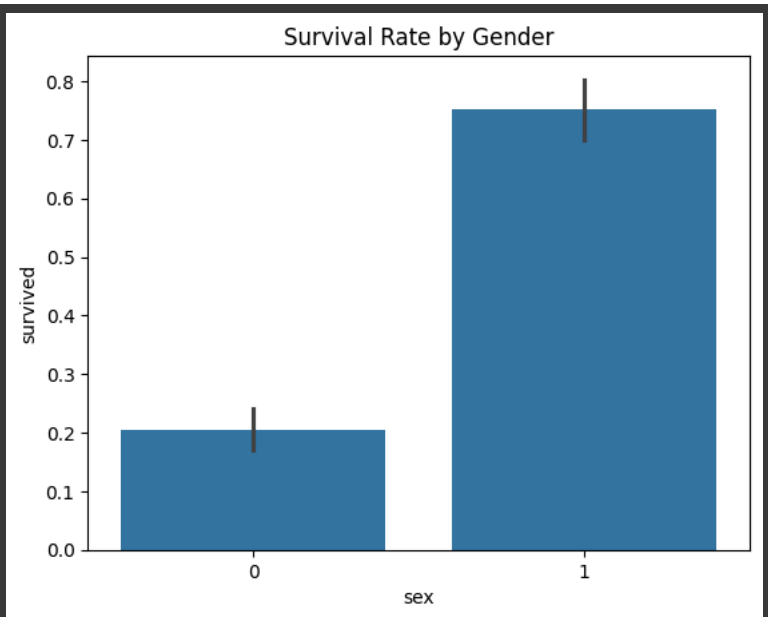
Explanation:

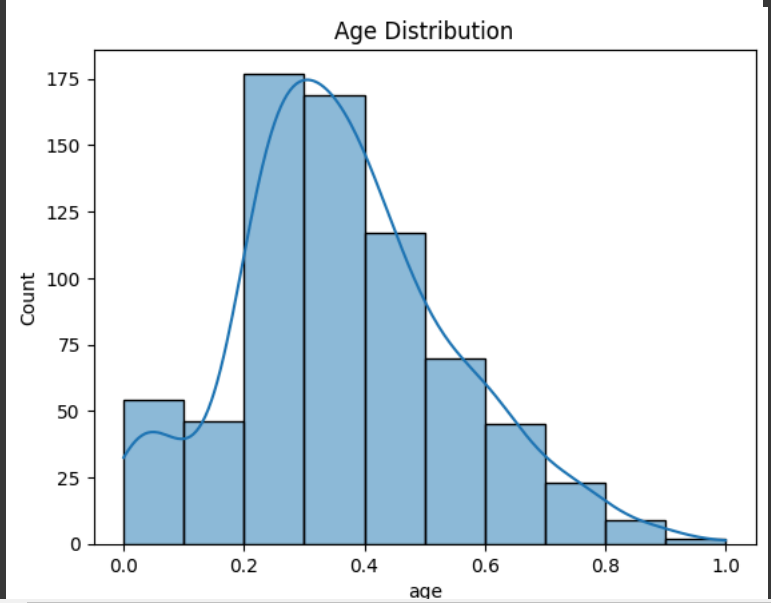
EDA helps visualize and summarize data to identify trends and patterns.

```
import matplotlib.pyplot as plt

# Survival rate by gender
sns.barplot(x='sex', y='survived', data=df_cleaned)
plt.title("Survival Rate by Gender")
plt.show()

# Distribution of age
sns.histplot(df_cleaned['age'], bins=10, kde=True)
plt.title("Age Distribution")
plt.show()
```





Output Explanation:

The first plot shows survival rates by gender.

The second plot shows the age distribution in the dataset.

8.Build the Models

Explanation: Build predictive models to classify or predict data.

```
# Example: Random Forest Model
from sklearn.ensemble import RandomForestClassifier

# Features and target
X = df_cleaned[['age', 'fare']]
y = df_cleaned['survived']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

# Evaluate the model
accuracy = rf_model.score(X_test, y_test)
print("Random Forest Model Accuracy:", accuracy)
```

```
Random Forest Model Accuracy: 0.6153846153846154
```

Output Explanation:

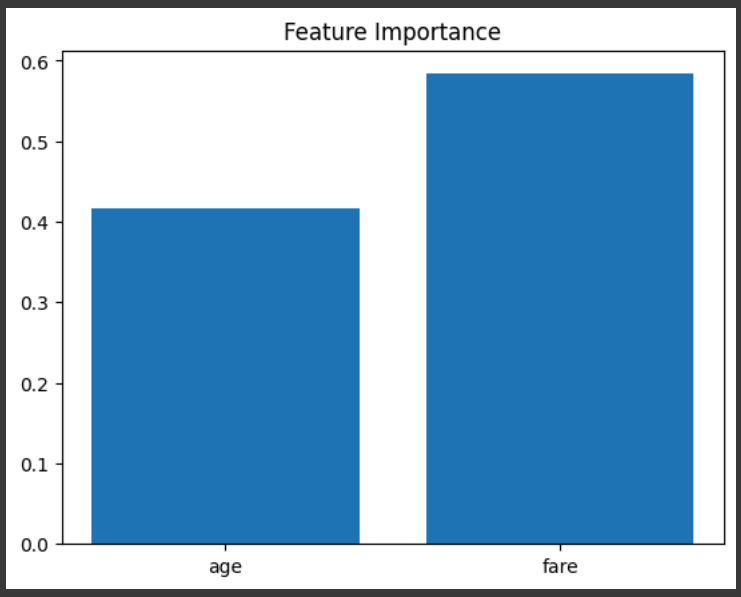
The Random Forest model predicts survival, with an accuracy score shown.

9.Presenting and Building Applications

Explanation: Present findings using visualizations or deploy the model into an application.

```
# Example: Visualize feature importance in Random Forest
importances = rf_model.feature_importances_
features = ['age', 'fare']

plt.bar(features, importances)
plt.title("Feature Importance")
plt.show()
```



Output Explanation:

Bar chart highlights which features (e.g., Age, Fare) are most important in predicting survival.

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```