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-1-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: Import necessary packages and load the dataset
import seaborn as sns
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
df = sns.load_dataset('titanic')
print("Preview of Data:\n", df.head())

# Step 2: Data preparation - Remove missing values
df_cleaned = df.dropna(subset=['age', 'fare', 'survived'])
print("\nAfter Cleaning:\n", df_cleaned.head())

# Step 3: Feature selection
X = df_cleaned[['age', 'fare']]
y = df_cleaned['survived']
print("\nSelected Features (X):\n", X.head())
print("\nTarget Variable (y):\n", y.head())

# Step 4: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("\nX_train Sample:\n", X_train.head())
print("\nX_test Sample:\n", X_test.head())
print("\ny_train Sample:\n", y_train.head())
print("\ny_test Sample:\n", y_test.head())

# Step 5: Train the model
model = LogisticRegression()
print("\nLogistic Regression Model Initialized.")

# Step 6: Fit the model
model.fit(X_train, y_train)
print("\nModel Training Completed.")

# Step 7: Evaluate the model
```

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

```
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

Selected Features (X):
     age     fare
0  22.0   7.2500
1  38.0  71.2833
2  26.0   7.9250
3  35.0  53.1000
4  35.0   8.0500

Target Variable (y):
 0    0
1    1
2    1
3    1
4    0
Name: survived, dtype: int64

X_train Sample:
       age     fare
328  31.0  20.5250
73   26.0  14.4542
253  30.0  16.1000
719  33.0   7.7750
666  25.0  13.0000

X_test Sample:
       age   fare
149  42.0  13.00
407   3.0  18.75
53   29.0  26.00
369  24.0  69.30
818  43.0   6.45

y_train Sample:
 328    1
73     0
253    0
719    0
666    0
Name: survived, dtype: int64

y_test Sample:
 149    0
407    1
53     1
369    1
818    0
Name: survived, dtype: int64

Logistic Regression Model Initialized.

Model Training Completed.

Model Accuracy: 0.6293706293706294
```

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.

```
# Step 1: Define the Research Goal
print("Step 1: Setting the Research Goal")
print("=" * 50)

# Problem Statement
problem_statement = """
The goal of this analysis is to predict the survival of Titanic passengers based on
important features such as Age, Gender, and Passenger Class. By understanding these factors,
we can analyze survival trends and build a predictive model for future similar scenarios.
"""
print(problem_statement)

# Define key research questions
research_questions = [
```

```
    "1. How does age influence the survival chances of a passenger?",
    "2. Does gender play a crucial role in survival rates?",
    "3. What is the impact of passenger class (1st, 2nd, 3rd) on survival probability?",
    "4. Can we develop a machine learning model to predict survival based on these features?"
]

print("Key Research Questions:")
for question in research_questions:
    print(question)

# Summary of Features Considered
features_considered = {
    "Age": "Numerical feature indicating passenger's age.",
    "Sex": "Categorical feature representing male or female.",
    "Pclass": "Categorical feature (1st, 2nd, 3rd class) showing socio-economic status."
}

print("\nFeatures Considered for Analysis:")
for feature, description in features_considered.items():
    print(f"- {feature}: {description}")

# Hypothesis Statement
print("\nHypothesis:")
print("Passengers who are younger, female, and from higher-class cabins have a higher survival rate.")

# Conclusion
print("\nBy setting this research goal, we establish a clear direction for data analysis and modeling.")
```

```
Step 1: Setting the Research Goal
==================================================

The goal of this analysis is to predict the survival of Titanic passengers based on
important features such as Age, Gender, and Passenger Class. By understanding these factors,
we can analyze survival trends and build a predictive model for future similar scenarios.

Key Research Questions:
1. How does age influence the survival chances of a passenger?
2. Does gender play a crucial role in survival rates?
3. What is the impact of passenger class (1st, 2nd, 3rd) on survival probability?
4. Can we develop a machine learning model to predict survival based on these features?

Features Considered for Analysis:
- Age: Numerical feature indicating passenger's age.
- Sex: Categorical feature representing male or female.
- Pclass: Categorical feature (1st, 2nd, 3rd class) showing socio-economic status.

Hypothesis:
Passengers who are younger, female, and from higher-class cabins have a higher survival rate.

By setting this research goal, we establish a clear direction for data analysis and modeling.
```

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-5-e52fe13ef617>: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-5-e52fe13ef617>:10: 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)

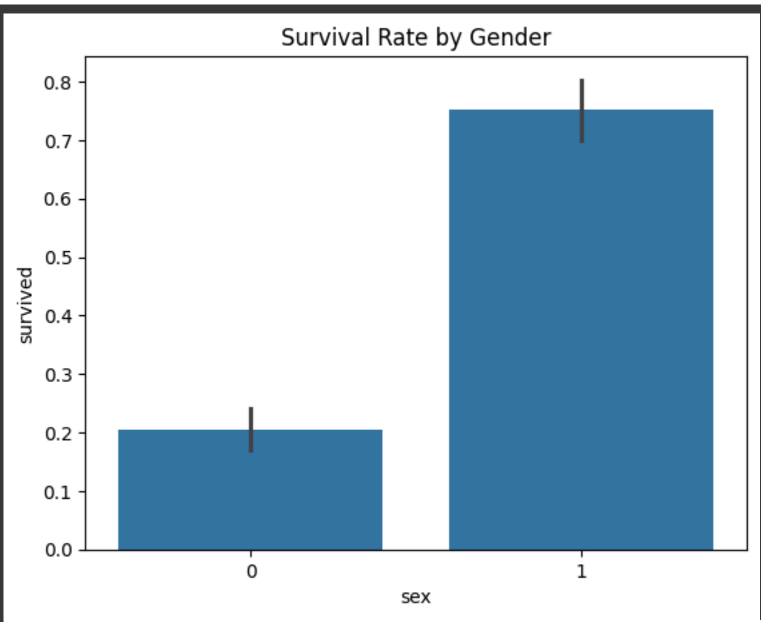
Explanation:

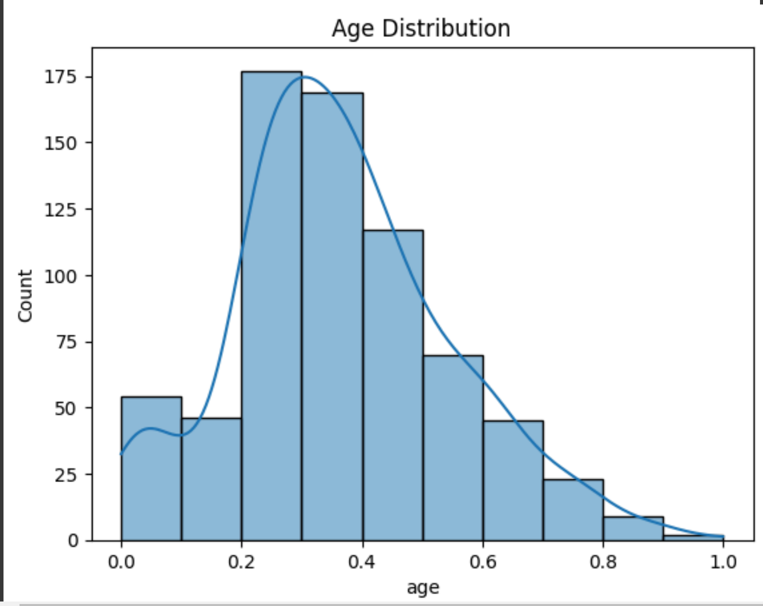
EDA helps visualize and summarize data to identify trends and patterns.

```
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.

```
# Step 1: Import Required Libraries
print("Step 1: Importing Required Libraries\n")
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd

print("✓ RandomForestClassifier and train_test_split imported successfully.\n")
print("=" * 60)

# Step 2: Prepare Features and Target Variable
print("Step 2: Defining Features (X) and Target (y)\n")

# Selecting relevant columns
X = df_cleaned[['age', 'fare']]
y = df_cleaned['survived']
print("Sample Features (X):\n", X.head(), "\n")
print("Sample Target (y):\n", y.head(), "\n")
print("✓ Features and Target selected successfully.\n")
print("=" * 60)

# Step 3: Train-Test Split
print("Step 3: Splitting Data into Training and Testing Sets\n")
```

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

print("Training Set - Features (X_train):\n", X_train.head(), "\n")
print("Training Set - Target (y_train):\n", y_train.head(), "\n")
print("Testing Set - Features (X_test):\n", X_test.head(), "\n")
print("Testing Set - Target (y_test):\n", y_test.head(), "\n")
print(f"✓ Train-Test Split Completed: {X_train.shape[0]} training samples, {X_test.shape[0]} testing samples.\n")
print("=" * 60)

# Step 4: Train the Random Forest Model
print("Step 4: Training the Random Forest Model\n")

# Initialize and train the model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

print("✓ Model Training Completed.")
print("Model Parameters:", rf_model.get_params(), "\n")
print("=" * 60)

# Step 5: Model Evaluation
print("Step 5: Evaluating the Model\n")

accuracy = rf_model.score(X_test, y_test)
print(f"✓ Random Forest Model Accuracy: {accuracy:.4f}\n")
print("=" * 60)
```

```
1    1
2    1
3    1
4    0
Name: survived, dtype: int64

✓ Features and Target selected successfully.

============================================================
Step 3: Splitting Data into Training and Testing Sets

Training Set - Features (X_train):
           age      fare
472  0.409399  0.054164
432  0.522493  0.050749
666  0.308872  0.025374
30   0.497361  0.054107
291  0.233476  0.177775

Training Set - Target (y_train):
 472    1
432    1
666    0
30     0
291    1
Name: survived, dtype: int64

Testing Set - Features (X_test):
           age      fare
641  0.296306  0.135265
496  0.673285  0.152766
262  0.648153  0.155466
311  0.220910  0.512122
551  0.334004  0.050749

Testing Set - Target (y_test):
 641    1
496    1
262    0
311    1
551    0
Name: survived, dtype: int64

✓ Train-Test Split Completed: 569 training samples, 143 testing samples.

============================================================
Step 4: Training the Random Forest Model

✓ Model Training Completed.
Model Parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_featur

============================================================
Step 5: Evaluating the Model

✓ Random Forest Model Accuracy: 0.5944

============================================================
```

Output Explanation:

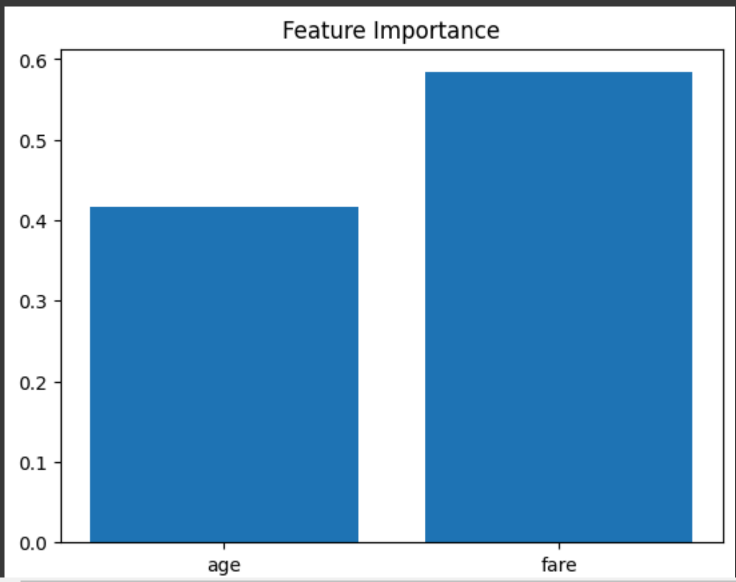
The Random Forest model predicts survival, with an accuracy score shown.

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.
```

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

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

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