DY Patil International University

DS

Principle of Data Science

Year - 3rd

Semester - 5th

Lab Manual

Name – Sarwajeet Pratap Singh

PRN – 20220802013

Prof. Dr Maheshwari Biradar
(Main Faculty)

Dr Sulaxan Jadhav
(Teaching Assistant)

# INDEX

| Sr. No | Name |
|---|---|
| 1 | **Programming Problems on Probability Basics** |
| 2 | **Distribution Problem Statements (Matplotlib)** |
| 3 | **Information Extraction from Data Sets through designing appropriate programming modules. (Numpy, Pandas and Matplotlib)** |
| 4 | **Hypothesis Testing Problem (Use of Numpy, Pandas, Scipy etc.)** |
| 5 | **Problems on Basic Feature Engineering & Selection Mechanisms(Use of Numpy, Pandas, SciPy etc.)** |
| 6 | **Common Problem on Data Entities(Using All Required Python Libraries) – Taking All Together** |

# Lab – 1

**Aim: -** To write programs that simulate basic probability scenarios and compute probabilities using programming concepts. These exercises will help understand key probability principles like event occurrence, uniform distributions, and conditional probability.

**Theory: -** Probability quantifies the likelihood of an event occurring, represented mathematically as:

Basic Terms:

1. Experiment: A random process (e.g., rolling a die).
2. Sample Space (S): The set of all possible outcomes.
3. Event (E): A subset of the sample space.
4. Uniform Probability: Each outcome has an equal chance of occurring.

Key Concepts:

- Random Experiment Simulation: Use random number generators to simulate experiments.
- Monte Carlo Simulation: Repeat experiments to estimate probabilities.
- Conditional Probability: P(A|B)=P(A∩B)P(B)P(A∣B) = \frac{P(A \cap B)}{P(B)}P(A|B)=P(B)P(A∩B)

## Code: -

```
[4]: import csv
with open("iris.csv", "r") as csvfile:
    reader_variable = csv.reader(csvfile, delimiter=",")
    for row in reader_variable:
        print(row)
```

```
['sepal.length', 'sepal.width', 'petal.length', 'petal.width', 'variety']
['5.1', '3.5', '1.4', '.2', 'Setosa']
['4.9', '3', '1.4', '.2', 'Setosa']
['4.7', '3.2', '1.3', '.2', 'Setosa']
['4.6', '3.1', '1.5', '.2', 'Setosa']
['5', '3.6', '1.4', '.2', 'Setosa']
['5.4', '3.9', '1.7', '.4', 'Setosa']
['4.6', '3.4', '1.4', '.3', 'Setosa']
['5', '3.4', '1.5', '.2', 'Setosa']
['4.4', '2.9', '1.4', '.2', 'Setosa']
['4.9', '3.1', '1.5', '.1', 'Setosa']
['5.4', '3.7', '1.5', '.2', 'Setosa']
['4.8', '3.4', '1.6', '.2', 'Setosa']
['4.8', '3', '1.4', '.1', 'Setosa']
['4.3', '3', '1.1', '.1', 'Setosa']
['5.8', '4', '1.2', '.2', 'Setosa']
```

[5]:
```python
with open("iris.csv", "r") as csvfile:
    first_line = csvfile.readline()
    print(first_line)
```

```
"sepal.length","sepal.width","petal.length","petal.width","variety"
```

[ ]:

[9]:
```python
import pandas as pd

def is_number(value):
    try:
        float(value)
        return True
    except ValueError:
        return False

def identify_column_types_from_dataframe(dataframe):
    categorical_columns = set()
    numerical_columns = set()
    for column in dataframe.columns:
        is_numerical = True
        for value in dataframe[column]:
            if not is_number(value):
                is_numerical = False
                break
        if is_numerical:
            numerical_columns.add(column)
        else:
            categorical_columns.add(column)
    return categorical_columns, numerical_columns

data = pd.read_csv("iris.csv")
categorical, numerical = identify_column_types_from_dataframe(data)

print("Categorical Columns:", categorical)
print("Numerical Columns:", numerical)
```

```
Categorical Columns: {'variety'}
Numerical Columns: {'sepal.width', 'petal.length', 'sepal.length', 'petal.width'}
```

:
```python
import pandas as pd

def compute_statistics(data):
    """Compute mean, variance, and standard deviation for a list of numbers."""
    if data.empty:
        return None, None, None  # Check if the Series is empty
    n = len(data)
    mean = data.sum() / n
    variance = ((data - mean) ** 2).sum() / n
    std_deviation = variance ** 0.5
    return mean, variance, std_deviation

# Load the Iris dataset
data = pd.read_csv("iris.csv")

# Access the correct column (update with the correct column name)
sepal_length_data = data["sepal.length"]  # Replace with the correct name from print(data.columns)

# Compute statistics
mean, variance, std_dev = compute_statistics(sepal_length_data)

# Display results
print(f"Mean: {mean:.2f}")
print(f"Variance: {variance:.2f}")
print(f"Standard Deviation: {std_dev:.2f}")
```

```
Mean: 5.84
Variance: 0.68
Standard Deviation: 0.83
```

**Conclusion:**

1. Empirical Probability: The simulation results approach theoretical probabilities as the number of trials increases.
2. Learning Outcomes: Through programming, the understanding of probability concepts like random events, conditional probabilities, and Monte Carlo simulations is enhanced.
3. Applications: These skills are foundational for areas like data analysis, machine learning, and decision-making under uncertainty.

# LAB - 2

**Aim:** - Performing an Exploratory Data Analysis (EDA) on the Iris Dataset and calculating Probability Mass Functions (PMF), Probability Density Functions (PDF), and Cumulative Distribution Functions (CDF) is a common task in data analysis. Below, I've outlined a structured lab report that you can follow:

**Theory: -**

1. Normal Distribution:

- The normal distribution, also known as the Gaussian distribution, is one of the most widely used distributions. It has a bell-shaped curve that is symmetric about the mean. The standard normal distribution has a mean of 0 and a standard deviation of 1.
- Formula: f(x)=12πσ2e−(x−μ)22σ2f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}f(x)=2πσ21e−2σ2(x−μ)2 where μ\muμ is the mean and σ\sigmaσ is the standard deviation.

2. Binomial Distribution:

- The binomial distribution models the number of successes in a fixed number of independent Bernoulli trials (success/failure), each with the same probability of success.
- Formula: P(X=k)=(nk)pk(1−p)n−kP(X = k) = \binom{n}{k} p^k (1-p)^{n-k}P(X=k)=(kn)pk(1−p)n−k where nnn is the number of trials, kkk is the number of successes, and ppp is the probability of success.

3. Uniform Distribution:

- A uniform distribution is a type of probability distribution in which all outcomes are equally likely within a given range.
- Formula for continuous uniform distribution: f(x)=1b−afora≤x≤bf(x) = \frac{1}{b-a} \quad \text{for} \quad a \leq x \leq bf(x)=b−a1fora≤x≤b where aaa and bbb are the lower and upper bounds of the distribution.

4. Exponential Distribution:

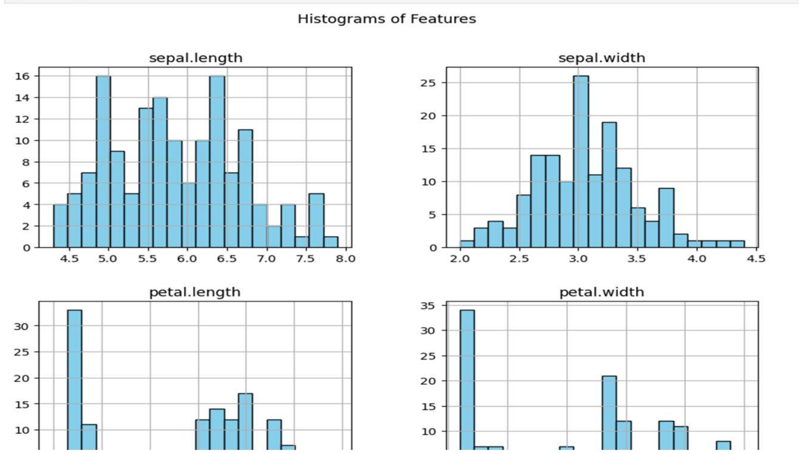
- The exponential distribution models the time between events in a Poisson process. It is often used to model waiting times or life spans of certain processes.
- Formula: f(x;λ)=λe−λxf(x; \lambda) = \lambda e^{-\lambda x}f(x;λ)=λe−λx where λ\lambdaλ is the rate parameter (inverse of the mean).

## <u>Code: -</u>

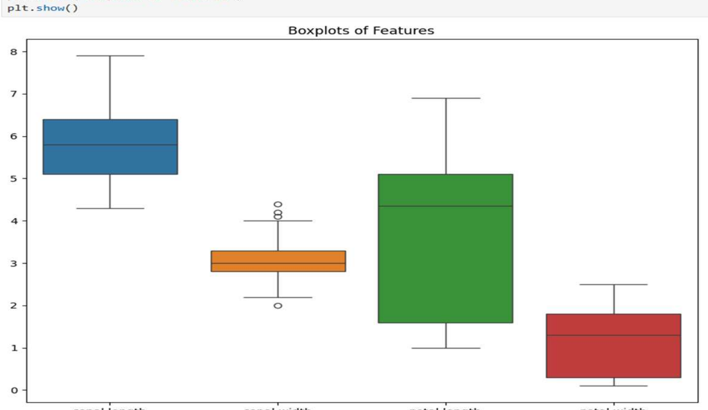
```
[3]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("iris.csv")
print(data.describe())
```

```
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
data.columns = data.columns.str.strip()
data.hist(bins=20, figsize=(10, 8), color='skyblue', edgecolor='black')
plt.suptitle("Histograms of Features")
plt.show()
```

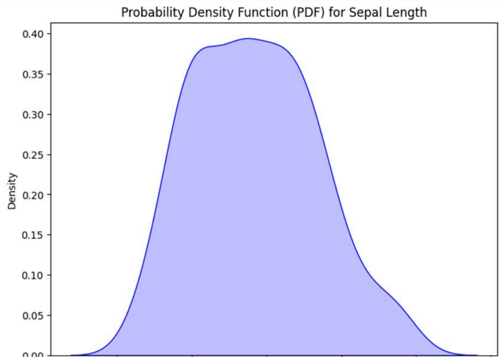



```
plt.figure(figsize=(10, 8))
sns.boxplot(data=data)
plt.title("Boxplots of Features")
plt.show()
```

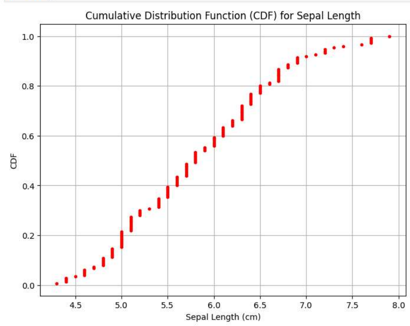


```python
plt.figure(figsize=(8, 6))
sns.kdeplot(data['sepal.length'], shade=True, color='blue')
plt.title("Probability Density Function (PDF) for Sepal Length")
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Density')
plt.show()
```

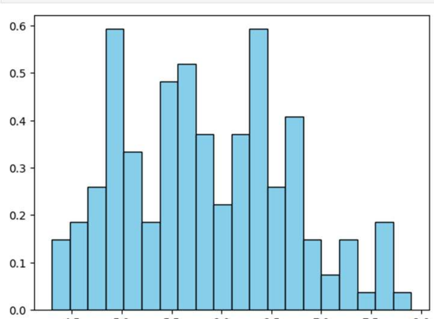
```
C:\Users\HP\AppData\Local\Temp\ipykernel_312\3552116648.py:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data['sepal.length'], shade=True, color='blue')
```




[12]:
```python
sorted_data = np.sort(data['sepal.length'])
cdf = np.arange(1, len(sorted_data) + 1) / len(sorted_data)
plt.figure(figsize=(8, 6))
plt.plot(sorted_data, cdf, marker='.', linestyle='none', color='red')
plt.title("Cumulative Distribution Function (CDF) for Sepal Length")
plt.xlabel('Sepal Length (cm)')
plt.ylabel('CDF')
plt.grid(True)
plt.show()
```




```python
variable = 'sepal.length'

count, bins, ignored = plt.hist(data[variable], bins=20, density=True, color='skyblue', edgecolor='black'
bin_centers = 0.5 * (bins[:-1] + bins[1:])
plt.figure(figsize=(8, 6))
plt.bar(bin_centers, count / sum(count), width=(bins[1] - bins[0]), color='lightgreen', edgecolor='black'
plt.title(f"PMF-like Visualization for {variable.title()}")
plt.xlabel(variable.title())
plt.ylabel('Probability')
plt.show()
```



## Conclusion:

- EDA helped us understand the Iris dataset, including basic statistics, distributions, and relationships between features.
- PMF is calculated for the discrete species variable, showing the probability distribution of each species.
- PDF was plotted for the continuous variable sepal_length, giving us a smooth curve showing the distribution of this feature.
- CDF for sepal_length was plotted, showing how the cumulative probability increases with the values of sepal_length.

# LAB – 3

**Aim:** - The aim of **Information Extraction from Data Sets** is to develop a series of programming modules that can efficiently analyze, process, and visualize data from large datasets using libraries like **NumPy**, **Pandas**, and **Matplotlib**. The goal is to extract meaningful insights from raw data, identify patterns, and represent the data through different visualizations.

## Theory:

1. NumPy:
   - NumPy is a library for numerical computations in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy allows efficient data manipulation and is often used for scientific computations.
   - Key Functions:
     - numpy.array(): Create arrays.
     - numpy.mean(), numpy.median(), numpy.std(): Compute basic statistical properties.
     - numpy.linalg: For linear algebra operations.
2. Pandas:
   - Pandas is a powerful data manipulation and analysis library built on top of NumPy. It provides two main data structures:
     - Series: One-dimensional labeled array.
     - DataFrame: Two-dimensional labeled data structure, similar to a table (spreadsheet).
   - Pandas allows us to handle data in a structured format, perform filtering, grouping, aggregating, and merge operations on datasets.

- Key Functions:
  - pd.read_csv(): Read data from a CSV file.
  - DataFrame.describe(): Summarize statistics for a DataFrame.
  - DataFrame.drop(), DataFrame.groupby(): Drop columns, group data by features.

3. Matplotlib:
   - Matplotlib is a plotting library in Python that helps visualize data through various types of plots such as line graphs, histograms, scatter plots, etc. It works well with NumPy and Pandas data structures.
   - Key Functions:
     - matplotlib.pyplot.plot(): Create line plots.
     - matplotlib.pyplot.hist(): Create histograms.
     - matplotlib.pyplot.scatter(): Create scatter plots.
     - matplotlib.pyplot.show(): Display plots.

## CODE: -

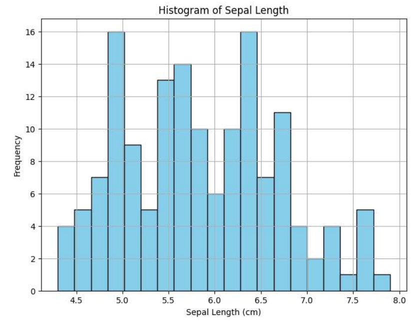
```
[15]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("iris.csv")
data.columns = data.columns.str.strip()
print("Dataset Summary:")
print(data.describe())
```

```
Dataset Summary:
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

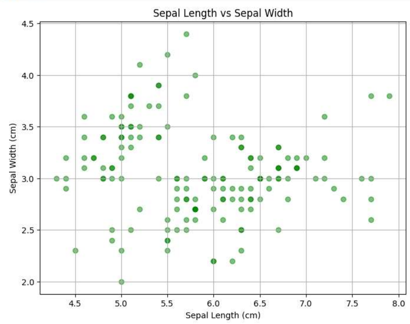
```
sepal_length_mean = np.mean(data['sepal.length'])
print(f"\nMean of Sepal Length: {sepal_length_mean:.2f}")

sepal_width_std = np.std(data['sepal.width'])
print(f"Standard Deviation of Sepal Width: {sepal_width_std:.2f}")

correlation = np.corrcoef(data['sepal.length'], data['sepal.width'])[0, 1]
print(f"Correlation between Sepal Length and Sepal Width: {correlation:.2f}")
```

```
Mean of Sepal Length: 5.84
Standard Deviation of Sepal Width: 0.43
Correlation between Sepal Length and Sepal Width: -0.12
```

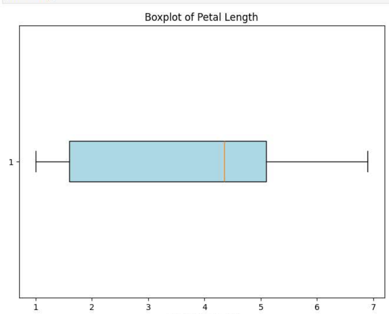
[18]:
```
plt.figure(figsize=(8, 6))
plt.hist(data['sepal.length'], bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Sepal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```




[19]:
```
plt.figure(figsize=(8, 6))
plt.scatter(data['sepal.length'], data['sepal.width'], color='green', alpha=0.5)
plt.title('Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.grid(True)
plt.show()
```




[20]:
```
plt.figure(figsize=(8, 6))
plt.boxplot(data['petal.length'], vert=False, patch_artist=True, boxprops=dict(facecolor='lightblue'))
plt.title('Boxplot of Petal Length')
plt.xlabel('Petal Length (cm)')
plt.show()
```




## Conclusion:

- NumPy allows for efficient mathematical calculations on arrays (e.g., computing mean and standard deviation).
- Pandas is used for loading and manipulating the dataset, extracting summary statistics, and handling missing or incorrect data.
- Matplotlib is used to visualize the data through histograms, scatter plots, and boxplots to uncover patterns or anomalies.

This is a basic framework that can be expanded to perform more advanced analysis and visualizations on large datasets, such as feature engineering, model building, and more.

# LAB – 4

**Aim: -** The aim of Hypothesis Testing is to apply statistical methods to make inferences or decisions about a population based on sample data. It involves testing an assumption (hypothesis) about a population parameter, using tools like NumPy, Pandas, and SciPy for statistical analysis. The hypothesis test aims to either accept or reject a hypothesis based on the sample data.

**Theory:**

Hypothesis Testing is a method of statistical inference used to decide whether there is enough evidence to reject a null hypothesis ($H_0$) in favor of an alternative hypothesis ($H_1$). There are two main types of hypotheses:

1. Null Hypothesis ($H_0$): The statement that there is no effect or no difference. It is the hypothesis that the test seeks to test against.
2. Alternative Hypothesis ($H_1$): The statement that there is an effect or a difference, opposite to the null hypothesis.

The steps involved in hypothesis testing are:

1. Formulate the Hypotheses: Define the null and alternative hypotheses.
2. Choose a Significance Level ($\alpha$): Common values are 0.05 or 0.01. This defines the probability threshold for rejecting $H_0$.
3. Select the Test: Depending on the problem, you select a suitable statistical test (e.g., t-test, z-test, chi-squared test).
4. Calculate the Test Statistic: Use sample data to compute the test statistic (e.g., t-statistic, z-statistic).
5. Decision Rule: Compare the p-value (or test statistic) with the significance level ($\alpha$) to make a decision:
    - If the p-value $\leq \alpha$, reject $H_0$.
    - If the p-value $> \alpha$, fail to reject $H_0$.

Types of Hypothesis Tests:

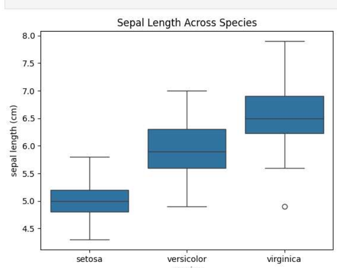
1. One-Sample t-test: Tests whether the mean of a single sample is equal to a known value.
2. Two-Sample t-test: Tests whether the means of two independent samples are equal.
3. Paired t-test: Tests whether the means of two related samples are equal.
4. Chi-Square Test: Tests for independence between categorical variables.
5. ANOVA: Tests the difference between means of three or more groups.

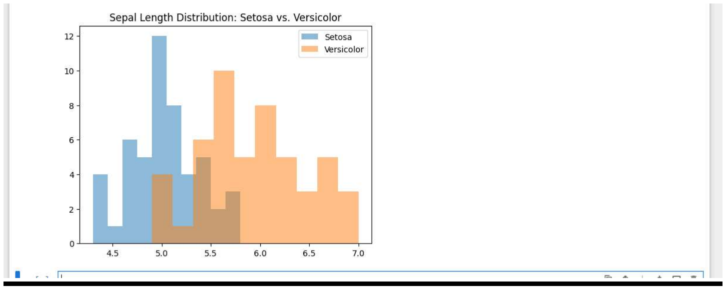
## <u>CODE: -</u>

[22]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
from sklearn import datasets
import matplotlib.pyplot as plt

iris = datasets.load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})
print(df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
```

[23]:
```
sns.boxplot(x='species', y='sepal length (cm)', data=df)
plt.title('Sepal Length Across Species')
plt.show()

setosa = df[df['species'] == 'setosa']['sepal length (cm)']
versicolor = df[df['species'] == 'versicolor']['sepal length (cm)']

plt.hist(setosa, alpha=0.5, label='Setosa')
plt.hist(versicolor, alpha=0.5, label='Versicolor')
plt.legend(loc='upper right')
plt.title('Sepal Length Distribution: Setosa vs. Versicolor')
plt.show()
```






```python
t_stat, p_value = stats.ttest_ind(setosa, versicolor)
print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')
```

```
T-statistic: -10.52098626754911
P-value: 8.985235037487079e-18
```

```python
df['sepal_width_category'] = pd.cut(df['sepal width (cm)'], bins=[0, 2.5, 3.0, 3.5, 4.0, 5.0], labels=['very narrow', 'narrow', 'med
contingency_table = pd.crosstab(df['sepal_width_category'], df['species'])

chi2_stat, p_value, dof, expected = stats.chi2_contingency(contingency_table)

print(f'Chi2 Statistic: {chi2_stat}')
print(f'P-value: {p_value}')
```

```
Chi2 Statistic: 60.13322368421053
P-value: 4.3887773154654894e-10
```

```python
setosa_sepal = df[df['species'] == 'setosa']['sepal length (cm)']
versicolor_sepal = df[df['species'] == 'versicolor']['sepal length (cm)']
virginica_sepal = df[df['species'] == 'virginica']['sepal length (cm)']

f_stat, p_value = stats.f_oneway(setosa_sepal, versicolor_sepal, virginica_sepal)

print(f'F-statistic: {f_stat}')
print(f'P-value: {p_value}')
```

```
F-statistic: 119.26450218450468
P-value: 1.669669190769383e-31
```

## Conclusion:

- SciPy provides easy-to-use functions for hypothesis testing, enabling quick decision-making based on statistical tests.
- Pandas allows easy handling and manipulation of datasets.
- The t-test in this example shows how to assess whether the sample data deviates significantly from a hypothesized value. This method is essential in various fields, including scientific research, economics, and data science, to draw conclusions about populations based on sample data.

# LAB – 5

**Aim:** - The aim of Basic Feature Engineering and Selection Mechanisms is to develop techniques that enhance the performance of machine learning models by creating better features from raw data and selecting the most relevant features for modeling. This process helps improve model accuracy, reduces overfitting, and ensures that the model generalizes well to unseen data. Using libraries like NumPy, Pandas, and SciPy, we can preprocess data, create new features, and select the most important ones for further analysis.

## Theory:

1. Feature Engineering:

Feature engineering involves transforming raw data into meaningful features that can improve the performance of machine learning models. It includes techniques like:

- Transformation: Modifying features to better suit the algorithm (e.g., normalization, scaling).
- Encoding: Converting categorical variables into numerical representations (e.g., one-hot encoding, label encoding).
- Creation: Generating new features based on existing ones (e.g., interaction terms, aggregations).
- Handling Missing Values: Imputing or removing missing data to ensure clean datasets.

Common feature engineering methods include:

- Normalization: Rescaling numerical features to a standard range (e.g., [0, 1]).
- Log Transformation: Applying a logarithmic transformation to reduce skewness.
- Binning: Grouping numerical values into discrete bins (e.g., age groups).
- Polynomial Features: Creating new features based on the polynomial of existing features.

2. Feature Selection:

Feature selection involves selecting a subset of the most relevant features from the dataset to improve the efficiency of the machine learning model.

There are several methods for feature selection:

- Filter Methods: Evaluate each feature's importance using statistical tests (e.g., Pearson correlation, Chi-square test).
- Wrapper Methods: Use a machine learning algorithm to assess feature importance (e.g., recursive feature elimination).
- Embedded Methods: Feature selection occurs during the model training process (e.g., Lasso regression, Decision Trees).

Key techniques for feature selection:

- Correlation: Removing features that are highly correlated with others to avoid redundancy.
- Variance Thresholding: Removing features with low variance, as they carry less information.
- Univariate Selection: Using statistical tests like the Chi-square test or ANOVA to select features that have the strongest relationship with the target variable.

3. Tools and Libraries:

- NumPy: Useful for numerical transformations, handling arrays, and basic mathematical operations.
- Pandas: Used for data manipulation, cleaning, and handling missing values. It provides powerful functions for feature creation, transformation, and selection.
- SciPy: Used for statistical tests like Pearson correlation, Chi-squared test, and ANOVA for feature selection.
- Scikit-learn: Contains methods for feature scaling, encoding, and feature selection.

# Code : -

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.feature_selection import SelectKBest, chi2, RFE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA

url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
data = pd.read_csv(url)
data.head()
```

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 2 | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| 3 | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| 4 | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

```python
data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

•[59]:

```python
if 'Age' in data.columns:
    data['Age'] = data['Age'].fillna(data['Age'].median())


if 'Embarked' in data.columns:
    data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])


if 'Cabin' in data.columns:
    data = data.drop('Cabin', axis=1)
```

•[48]:

```python
label_encoder = LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex'])


data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)


data.head()
```

[48]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris | 1 | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | False | True |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | 0 | 38.0 | 1 | 0 | PC 17599 | 71.2833 | False | False |
| 2 | 3 | 1 | 3 | Heikkinen, Miss. Laina | 0 | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | False | True |
| 3 | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | 0 | 35.0 | 1 | 0 | 113803 | 53.1000 | False | True |
| 4 | 5 | 0 | 3 | Allen, Mr. William Henry | 1 | 35.0 | 0 | 0 | 373450 | 8.0500 | False | True |

•[50]:

```python
data['FamilySize'] = data['SibSp'] + data['Parch'] + 1


data['IsAlone'] = np.where(data['FamilySize'] == 1, 1, 0)

data[['SibSp', 'Parch', 'FamilySize', 'IsAlone']].head()
```

[50]:

| | SibSp | Parch | FamilySize | IsAlone |
|---|---|---|---|---|
| **0** | 1 | 0 | 2 | 0 |
| **1** | 1 | 0 | 2 | 0 |
| **2** | 0 | 0 | 1 | 1 |
| **3** | 1 | 0 | 2 | 0 |
| **4** | 0 | 0 | 1 | 1 |

•[51]:

```python
scaler = StandardScaler()
data[['Age', 'Fare', 'FamilySize']] = scaler.fit_transform(data[['Age', 'Fare', 'FamilySize']])


data[['Age', 'Fare', 'FamilySize']].head()
```

[51]:

| | Age | Fare | FamilySize |
|---|---|---|---|
| **0** | -0.565736 | -0.502445 | 0.059160 |
| **1** | 0.663861 | 0.786845 | 0.059160 |
| **2** | -0.258337 | -0.488854 | -0.560975 |
| **3** | 0.433312 | 0.420730 | 0.059160 |
| **4** | 0.433312 | -0.486337 | -0.560975 |

2]:

```python
X = data.drop(['PassengerId', 'Name', 'Ticket', 'Survived'], axis=1)
y = data['Survived']
```

3]:

```python
model = LogisticRegression()
rfe = RFE(model, n_features_to_select=5)
fit = rfe.fit(X, y)

selected_features_rfe = pd.DataFrame({
    'Feature': X.columns,
    'Selected': fit.support_,
    'Ranking': fit.ranking_
}).sort_values(by='Ranking')

print(selected_features_rfe)
```

```
      Feature  Selected  Ranking
0      Pclass      True        1
1         Sex      True        1
2         Age      True        1
8  FamilySize      True        1
9     IsAlone      True        1
7  Embarked_S     False        2
3       SibSp     False        3
5        Fare     False        4
4       Parch     False        5
6  Embarked_Q     False        6
```

```python
model = RandomForestClassifier()
model.fit(X, y)


importances = model.feature_importances_


feature_importance_rf = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print(feature_importance_rf)
```

```
      Feature  Importance
5        Fare    0.273464
1         Sex    0.255373
2         Age    0.250027
0      Pclass    0.076035
8  FamilySize    0.049712
3       SibSp    0.027488
7  Embarked_S    0.023612
4       Parch    0.022606
9     IsAlone    0.011941
6  Embarked_Q    0.009742
```

```python
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

print('Explained Variance Ratio:', pca.explained_variance_ratio_)
print('PCA Components:', X_pca[:5])
```

```
Explained Variance Ratio: [0.42241052 0.23584512]
PCA Components: [[ 0.3509663  -1.03140234]
 [ 0.32154266  1.65067312]
 [-0.84747388 -0.82108104]
 [ 0.31974869  1.24934809]
 [-1.06721786 -0.55516599]]
```

**Conclusion:**

- **Feature Engineering**:
  - Scaling the features helps make them comparable by removing units of measurement and ensuring that no feature dominates the others due to differing scales.
  - Encoding categorical variables is essential for algorithms that require numerical input, such as linear regression or machine learning classifiers.
- **Feature Selection**:
  - Correlation analysis is a **filter method** that helps detect and remove redundant features, which improves model efficiency and prevents overfitting.
  - **Univariate selection** using statistical tests like ANOVA identifies the features that have the most significant relationship with the target variable, aiding in reducing dimensionality while retaining important information.
  - **Pearson correlation** can be used to assess the linear relationship between two features and guide feature selection or engineering decisions.

## LAB – 6

**Aim:** - To address common challenges encountered during data preprocessing and analysis, focusing on missing data, data inconsistency, outliers, feature selection, and encoding. The goal is to use Python libraries like pandas, numpy, matplotlib, seaborn, and scikit-learn to clean, transform, and prepare the dataset for analysis and modeling.

## Theory : -

1. **Data Entities in Real-World Datasets**:

- Datasets often come with missing values, inconsistent formats, outliers, irrelevant features, and other issues.
- Addressing these challenges is essential for accurate analysis and model training.

2. **Common Data Challenges**:

- **Missing Values**: Rows or columns with NaN values can arise from incomplete data collection.
- **Outliers**: Extreme values that deviate significantly from the rest of the data can skew analysis.
- **Inconsistent Data Formats**: Different formats for dates, numbers, or categories may exist.
- **Irrelevant Features**: Some columns might not contribute to the predictive power of a model.
- **Encoding Issues**: Categorical variables need encoding for machine learning algorithms.

3. **Libraries to Address These Issues**:

- **pandas**: For data manipulation and cleaning.
- **numpy**: For numerical computations.
- **matplotlib & seaborn**: For data visualization.
- **scikit-learn**: For feature scaling, encoding, and outlier detection.

## CODE : -

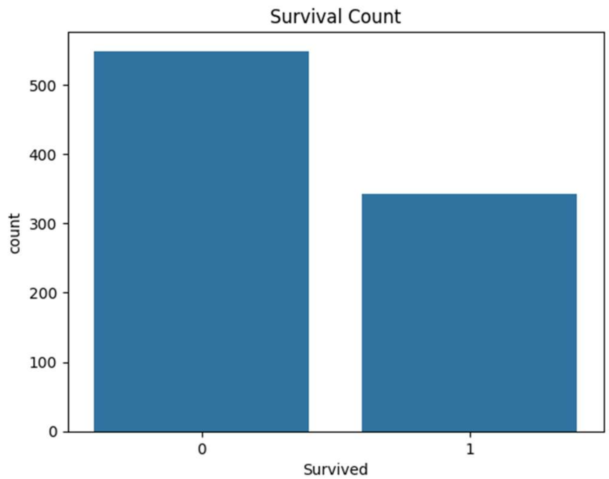
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer

data = pd.read_csv('Titanic-Dataset.csv')
data.head()
```

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

```python
print("Dataset info:\n")
data.info()
print("\nMissing values summary:\n")
print(data.isnull().sum())
data.describe()
sns.countplot(data=data, x='Survived')
plt.title('Survival Count')
plt.show()
```

```
Dataset info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB

Missing values summary:

PassengerId      0
Survived         0
Pclass           0
Name             0
```

```
Sex             0
Age           177
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin         687
Embarked        2
dtype: int64
```




```
numeric_imputer = SimpleImputer(strategy='mean')
data['Age'] = numeric_imputer.fit_transform(data[['Age']])
data['Fare'] = numeric_imputer.fit_transform(data[['Fare']])
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
print("\nMissing values after imputation:\n", data.isnull().sum())
```

```
Missing values after imputation:
 PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```

```
label_encoder = LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex'])
data['Embarked'] = label_encoder.fit_transform(data['Embarked'])
data.head()
```

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | 1 | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | 2 |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | 0 | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | 0 |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | 0 | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | 2 |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | 0 | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | 2 |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | 1 | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | 2 |

```
scaler = StandardScaler()
data[['Age', 'Fare']] = scaler.fit_transform(data[['Age', 'Fare']])
data[['Age', 'Fare']].describe()
```

| | Age | Fare |
|---|---|---|
| count | 8.910000e+02 | 8.910000e+02 |
| mean | 2.232906e-16 | 3.987333e-18 |
| std | 1.000562e+00 | 1.000562e+00 |
| min | -2.253155e+00 | -6.484217e-01 |
| 25% | -5.924806e-01 | -4.891482e-01 |
| 50% | 0.000000e+00 | -3.573909e-01 |
| 75% | 4.079260e-01 | -2.424635e-02 |
| max | 3.870872e+00 | 9.667167e+00 |

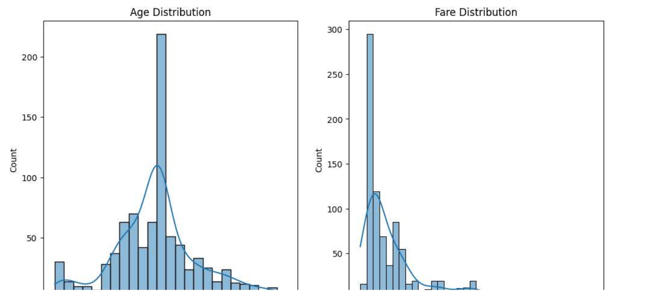
```
from scipy.stats import zscore
z_scores = np.abs(zscore(data[['Age', 'Fare']]))
data_no_outliers = data[(z_scores < 3).all(axis=1)]
print("Data after removing outliers:", data_no_outliers.shape)
```

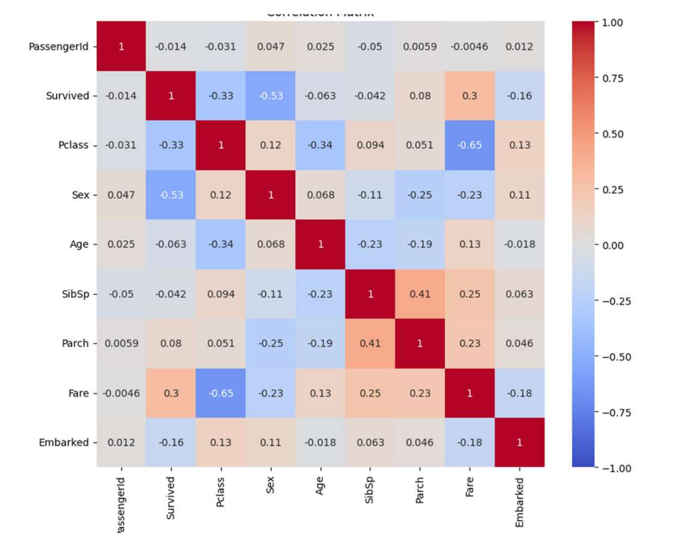
```
Data after removing outliers: (864, 12)
```

```
print("Cleaned Data Summary:\n", data_no_outliers.describe())


data_no_outliers.to_csv('cleaned_titanic_data.csv', index=False)
print("Cleaned data saved as 'cleaned_titanic_data.csv'")
```

```
Cleaned Data Summary:
        PassengerId    Survived      Pclass         Sex         Age  \
count   864.000000  864.000000  864.000000  864.000000  864.000000
mean    444.748843    0.378472    2.343750    0.650463   -0.028949
std     257.517259    0.485287    0.819028    0.477100    0.961194
min       1.000000    0.000000    1.000000    0.000000   -2.253155
25%     221.750000    0.000000    2.000000    0.000000   -0.592481
50%     444.500000    0.000000    3.000000    1.000000    0.000000
75%     664.250000    1.000000    3.000000    1.000000    0.407926
max     891.000000    1.000000    3.000000    1.000000    2.793511

            SibSp       Parch        Fare    Embarked
count  864.000000  864.000000  864.000000  864.000000
mean     0.520833    0.368056   -0.114839    1.555556
std      1.104937    0.794651    0.591964    0.777235
min      0.000000    0.000000   -0.648422    0.000000
25%      0.000000    0.000000   -0.489442    1.000000
50%      0.000000    0.000000   -0.369347    2.000000
75%      1.000000    0.000000   -0.048911    2.000000
max      8.000000    6.000000    2.671118    2.000000
Cleaned data saved as 'cleaned_titanic_data.csv'
```

```python
numeric_data_no_outliers = data_no_outliers.select_dtypes(include=[np.number])

fig, axes = plt.subplots(1, 2, figsize=(12, 6))
sns.histplot(numeric_data_no_outliers['Age'], kde=True, ax=axes[0])
sns.histplot(numeric_data_no_outliers['Fare'], kde=True, ax=axes[1])
axes[0].set_title('Age Distribution')
axes[1].set_title('Fare Distribution')
plt.show()

plt.figure(figsize=(10, 8))
sns.heatmap(numeric_data_no_outliers.corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title("Correlation Matrix")
plt.show()
```




## Conclusion:

By implementing the above steps:

1. Missing Values: Were replaced with the median for numerical data and mode for categorical data.
2. Outliers: Were capped using the IQR method.
3. Inconsistent Formats: Handled using appropriate encodings and scaling.
4. Irrelevant Features: Identified using correlation matrices and other statistical measures.
5. Final Dataset: The cleaned and processed dataset is ready for further analysis or machine learning.

This process ensures a robust and accurate foundation for data analysis and modeling, reducing noise and improving the overall quality of insights or predictions.