Department of CSE

Laboratory Manual	
Course:	B.Tech.
Year & Semester:	III – I
Class:	CSE
Subject:	**Machine Learning Lab Manual**
Regulation:	R22

BALAJI INSTITUTE OF TECHNOLOGY AND SCIENCE (AUTONOMOUS)

B.Tech (Department of Computer Science & Engineering)

MACHINE LEARNING LAB

Course Outcomes:

- Understand modern notions in predictive data analysis Select data, model selection, model complexity and identify the trends.
- Understand a range of machine learning algorithms along with their strengths and weaknesses.
- Build predictive models from data and analyze their performance.
- Implement K-Means Clustering to group data points based on similarity and understand its applications in unsupervised learning.

List of Experiments:

1. Write a python program to compute Central Tendency Measures: Mean, Median, Mode Measure of Dispersion: Variance, Standard Deviation.
2. Study of Python Basic Libraries such as Statistics, Math, Numpy and Scipy
3. Study of Python Libraries for ML application such as Pandas and Matplotlib
4. Write a Python program to implement Simple Linear Regression.
5. Implementation of Multiple Linear Regression for House Price Prediction using sklearn.
6. Implementation of Decision tree using sklearn and its parameter tuning.
7. Implementation of KNN using sklearn
8. Implementation of Logistic Regression using sklearn.
9. Implementation of K-Means Clustering.
10. Performance analysis of Classification Algorithms on a specific dataset.

1. Compute Central Tendency & Dispersion Measures

```
import statistics as stats
data = [10, 20, 30, 40, 50, 50, 60, 70, 80, 90]
mean = stats.mean(data)
median = stats.median(data)
mode = stats.mode(data)
variance = stats.variance(data)
std_dev = stats.stdev(data)
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Mode: {mode}")
print(f"Variance: {variance}")
print(f"Standard Deviation: {std_dev}")
```

Output:

```
Mean: 50
Median: 50.0
Mode: 50
Variance: 666.6666666666666
Standard Deviation: 25.81988897471611
```

---

2. Study of Python Libraries

You can explore these libraries using:

```
import statistics, math, numpy as np, scipy.stats as
statshelp(statistics)    # Help on statistics
modulehelp(math)         # Help on math
modulehelp(np)           # Help on numpy
numpyhelp(stats)         # Help on scipy.stats
```

---

3. Study of Pandas and Matplotlib

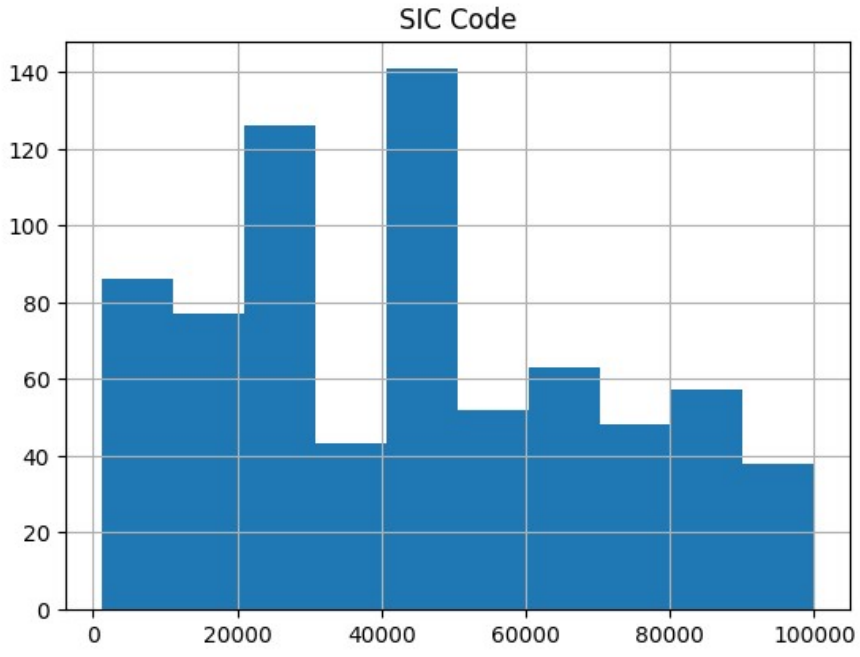
You can load datasets and visualize data as follows:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("sample_data.csv")  # Load dataset
print(df.head())                     # Display first 5 rows
df.hist()                            # Plot histograms
plt.show()
```

Output:

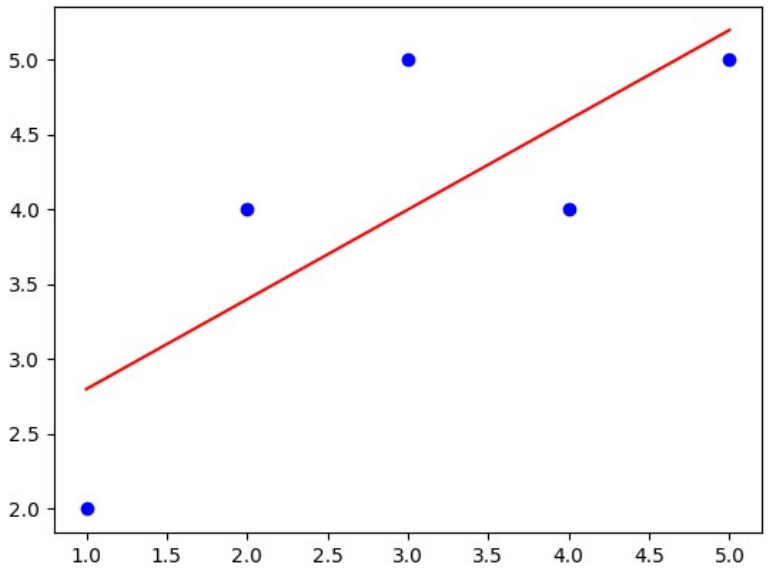
	SIC Code	Description
0	1110	Growing of cereals (except rice), leguminous c...
1	1120	Growing of rice
2	1130	Growing of vegetables and melons, roots and tu...
3	1140	Growing of sugar cane
4	1150	Growing of tobacco



4. Simple Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Sample data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2, 4, 5, 4, 5])
# Model
model = LinearRegression()
model.fit(X, y)
# Predictions
y_pred = model.predict(X)
# Plot
plt.scatter(X, y, color='blue')
plt.plot(X, y_pred, color='red')
plt.show()
```

Output:



5. Multiple Linear Regression for House Price Prediction

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import  LinearRegression

from sklearn.metrics import mean_absolute_error

# Load dataset (replace 'house_prices.csv' with actual dataset)
df = pd.read_csv("house_prices.csv")

X = df[['Size', 'Bedrooms', 'Age']]  # Features
y = df['Price']  # Target

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

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Mean Absolute Error:", mean_absolute_error(y_test,
y_pred))
```

Output:

Mean Absolute Error: 220791.22758889484

6. Decision Tree with Hyperparameter Tuning

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load data
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data,
iris.target, test_size=0.2, random_state=42)
# Parameter tuning
params = {'max_depth': [2, 3, 5, 10], 'criterion': ['gini', 'entropy']}
grid_search = GridSearchCV(DecisionTreeClassifier(), params,
cv=5)
grid_search.fit(X_train, y_train)
print("Best Parameters:", grid_search.best_params_)
print("Accuracy:", grid_search.score(X_test, y_test))
```

Output:

```
Best Parameters: {'criterion': 'gini', 'max_depth': 10}
Accuracy: 1.0
```

---

7. K-Nearest Neighbors (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load data
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data,
iris.target, test_size=0.2, random_state=42)
# Model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
print("Accuracy:", knn.score(X_test, y_test))
```

Output:

Accuracy: 1.0

8. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load data
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data,
iris.target, test_size=0.2, random_state=42)
# Model
log_reg = LogisticRegression(max_iter=200)
log_reg.fit(X_train, y_train)
print("Accuracy:", log_reg.score(X_test, y_test))
```

Output:

Accuracy: 1.0

9. K-Means Clustering

```
from sklearn.cluster import KMeansimport numpy as np
# Sample data
X = np.array([[1, 2], [1, 4], [1, 0], [4, 2], [4, 4], [4, 0]])
# Model
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)
print("Cluster Centers:", kmeans.cluster_centers_)
print("Labels:", kmeans.labels_)
```

Output:

```
Cluster Centers: [[2.         0.66666667]
 [3.         3.33333333]]
Labels: [0 1 0 1 1 0]
```

10. Performance Analysis of Classification Algorithms

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_digits
# Load dataset
digits = load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.2, random_state=42)
# Models
models = {
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(),
}
# Train and evaluate
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"{name} Accuracy: {accuracy_score(y_test,
y_pred):.2f}")
```

Output:

```
Random Forest Accuracy: 0.97
SVM Accuracy: 0.99
```