LAB CODE AND OUTPUTS

1. Implement and demonstratethe FIND-Salgorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

CODE:

```
import pandas as pd

# Load CSV
df = pd.read_csv("/content/1.csv")

hypothesis = None
print("Step | Instance        | Label | Hypothesis")
print("----------------------------------------------")

for step, (features, label) in enumerate(zip(df.iloc[:, :-1].values,
df.iloc[:, -1].str.lower()), 1):
    if label == 'yes':
        hypothesis = features if hypothesis is None else [
            '?' if h != f else h for h, f in zip(hypothesis, features)
        ]
        print(f"{step:<4}| {list(features)} | {label:<5}| {hypothesis}")
    else:
        print(f"{step:<4}| {list(features)} | {label:<5}| Ignored")

print("\nFinal Most Specific Hypothesis:", hypothesis)
```

OUTPUT:

```
Step | Instance        | Label | Hypothesis

----------------------------------------------
1   | ['high', 'good'] | yes   | ['high' 'good']
2   | ['medium', 'good'] | yes   | ['?', 'good']
3   | ['medium', 'poor'] | no    | Ignored
4   | ['low', 'good'] | no    | Ignored
5   | ['high', 'good'] | yes   | ['?', 'good']
6   | ['medium', 'good'] | yes   | ['?', 'good']
```

```
Final Most Specific Hypothesis: ['?', 'good']
```

2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the CandidateElimination algorithmto output a description of the set of all hypotheses consistent with the training examples.

CODE:

```
import pandas as pd

# Load Data
data = pd.read_csv("/content/2.csv").values.tolist()
n = len(data[0]) - 1
S = ['?'] * n
G = [['*'] * n]

# Candidate Elimination
for features, label in [(row[:-1], row[-1].lower()) for row in data]:
    if label == 'yes':  # Positive example
        G = [g for g in G if all(g[i] in ('*', features[i]) for i in
range(n))]
        S = [f if s == '?' else '*' if s != f else s for s, f in zip(S,
features)]
    else:  # Negative example
        new_G = []
        for g in G:
            for i in range(n):
                if g[i] == '*':
                    for val in {r[i] for r in data if r[-1].lower() ==
'yes'}:
                        if features[i] != val:
                            h = g.copy(); h[i] = val
                            new_G.append(h)
        G = new_G

G = [g for g in G if any(g[i] == '*' or g[i] == s for i, s in
enumerate(S))]

# Output
print("Final Specific Hypothesis (S):", S)
print("Final General Hypotheses (G):")
for g in G:
```

```
    print(g)
```

OUTPUT:

```
Final Specific Hypothesis (S): ['Sunny', 'Warm', '*', 'Strong', '*', '*']
Final General Hypotheses (G):
['Sunny', '*', '*', '*', '*', '*']
['*', 'Warm', '*', '*', '*', '*']
['*', '*', '*', '*', 'Cool', '*']
```

3. Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

CODE:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, export_text

# Load and encode data
df = pd.read_csv("/content/3.csv").apply(LabelEncoder().fit_transform)

# Train model
X, y = df.drop("Pass", axis=1), df["Pass"]
model = DecisionTreeClassifier(criterion="entropy").fit(X, y)

# Show tree and predict
print(export_text(model, feature_names=X.columns))
sample = pd.DataFrame([['medium', 'good']], columns=X.columns)
sample = sample.apply(LabelEncoder().fit_transform)
print("Predicted class:", "yes" if model.predict(sample)[0] else "no")
```

OUTPUT:

```
|--- Internal_Marks <= 0.50
|   |--- class: 1
|--- Internal_Marks >  0.50
|   |--- Internal_Marks <= 1.50
```

```
|   |   |--- class: 0
|   |--- Internal_Marks >  1.50
|   |   |--- Attendance <= 0.50
|   |   |   |--- class: 1
|   |   |--- Attendance >  0.50
|   |   |   |--- class: 0
```

Predicted class: yes

4. Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file.Compute the accuracy of the classifier, considering few test data sets.

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import CategoricalNB
from sklearn.metrics import accuracy_score

# Load and encode data
df = pd.read_csv("/content/4.csv").apply(LabelEncoder().fit_transform)
X, y = df.drop('Play', axis=1), df['Play']

# Train and test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Model training and prediction
model = CategoricalNB().fit(X_train, y_train)
y_pred = model.predict(X_test)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))
```

OUTPUT:

```
Accuracy: 0.8
```

5. Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.

CODE:

```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load data
df = pd.read_csv("/content/5.csv")
X, y = df.iloc[:, :4], df['target']

# Train model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = KNeighborsClassifier(n_neighbors=3).fit(X_train, y_train)
y_pred = model.predict(X_test)

# Output
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print("Correct:", sum(y_test == y_pred), " Wrong:", sum(y_test != y_pred))
```

OUTPUT:

```
Accuracy: 1.00
Correct: 30  Wrong: 0
```

6.Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

CODE:

```
import numpy as np, pandas as pd

# Load data
df = pd.read_csv("/content/6.csv")
X, y = df[['Input1', 'Input2']].values, df[['Output']].values

# Activation
sigmoid = lambda x: 1 / (1 + np.exp(-x))
sigmoid_deriv = lambda x: x * (1 - x)

# Initialize
np.random.seed(1)
wh, bh = np.random.rand(2, 2), np.random.rand(1, 2)
wo, bo = np.random.rand(2, 1), np.random.rand(1, 1)

# Training
for _ in range(10000):
    h = sigmoid(X @ wh + bh)
    out = sigmoid(h @ wo + bo)
    d_out = (y - out) * sigmoid_deriv(out)
    d_hidden = (d_out @ wo.T) * sigmoid_deriv(h)
    wo += h.T @ d_out * 0.1; bo += d_out.sum(0, keepdims=True) * 0.1
    wh += X.T @ d_hidden * 0.1; bh += d_hidden.sum(0, keepdims=True) * 0.1

# Output
print("Final Output:\n", out.round())
```

OUTPUT:

```
Final Output:
 [[0.]
 [1.]
 [1.]
 [0.]]
```

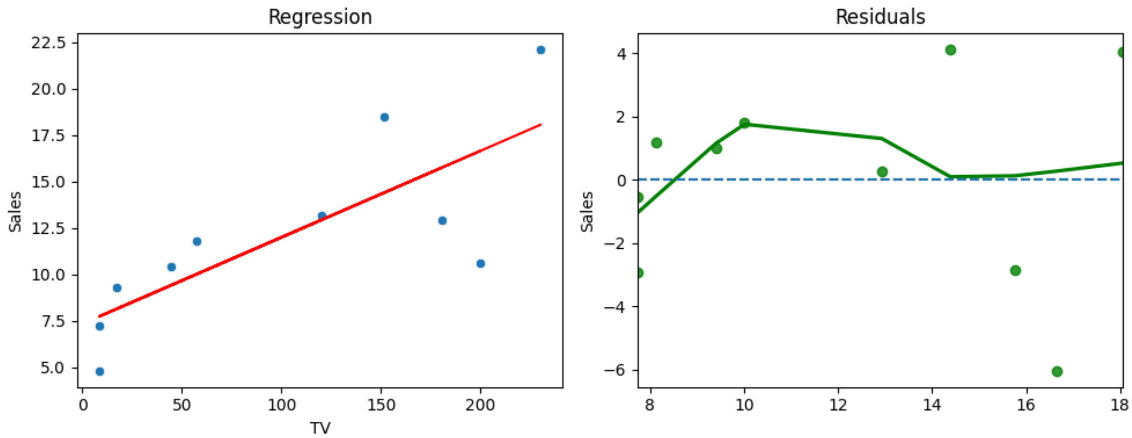
7. Write a program to demonstrate Regression analysis with residual plots on a given data set.

CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns

# Load and fit model
df = pd.read_csv("/content/7.csv")
X, y = df[['TV']], df['Sales']
model = LinearRegression().fit(X, y)
y_pred = model.predict(X)

print("Intercept:", model.intercept_, "Coefficient:", model.coef_[0])

# Plot regression and residuals
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
sns.scatterplot(x=X['TV'], y=y); plt.plot(X, y_pred, color='red');
plt.title("Regression")
plt.subplot(1, 2, 2)
sns.residplot(x=y_pred, y=y - y_pred, lowess=True, color='green');
plt.axhline(0, ls='--')
plt.title("Residuals"); plt.tight_layout(); plt.show()
```

OUTPUT:

Intercept: 7.33401842471077 Coefficient: 0.046579463885457154



8. Write a program to compute summary statistics such as mean, median, mode, standard deviationand variance of the given different types of data.

CODE:

```
import pandas as pd
from scipy import stats

# Load dataset
df = pd.read_csv('/content/8.csv')

# Summary statistics
print("Mean:\n", df.mean())
print("\nMedian:\n", df.median())
print("\nMode:\n", df.mode().iloc[0])
print("\nStandard Deviation:\n", df.std())
print("\nVariance:\n", df.var())
```

OUTPUT:

```
Mean:
 Study_Hours     5.5
Exam_Score     64.1
dtype: float64
```

```
Median:
 Study_Hours     5.5
Exam_Score     64.0
dtype: float64

Mode:
 Study_Hours     1
Exam_Score     50
Name: 0, dtype: int64

Standard Deviation:
 Study_Hours    3.027650
Exam_Score     9.620002
dtype: float64

Variance:
 Study_Hours     9.166667
Exam_Score     92.544444
dtype: float64
```

9. Write a program to implement k-Means clustering algorithm to cluster the set of data stored in .CSV file.

CODE:

```python
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load data
X = pd.read_csv('/content/9.csv').values

# K-Means clustering
model = KMeans(n_clusters=3, random_state=0).fit(X)
labels = model.labels_

# Plot
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:, 1],
            s=200, c='red', marker='X')
plt.title("K-Means Clustering"); plt.show()
```

OUTPUT: