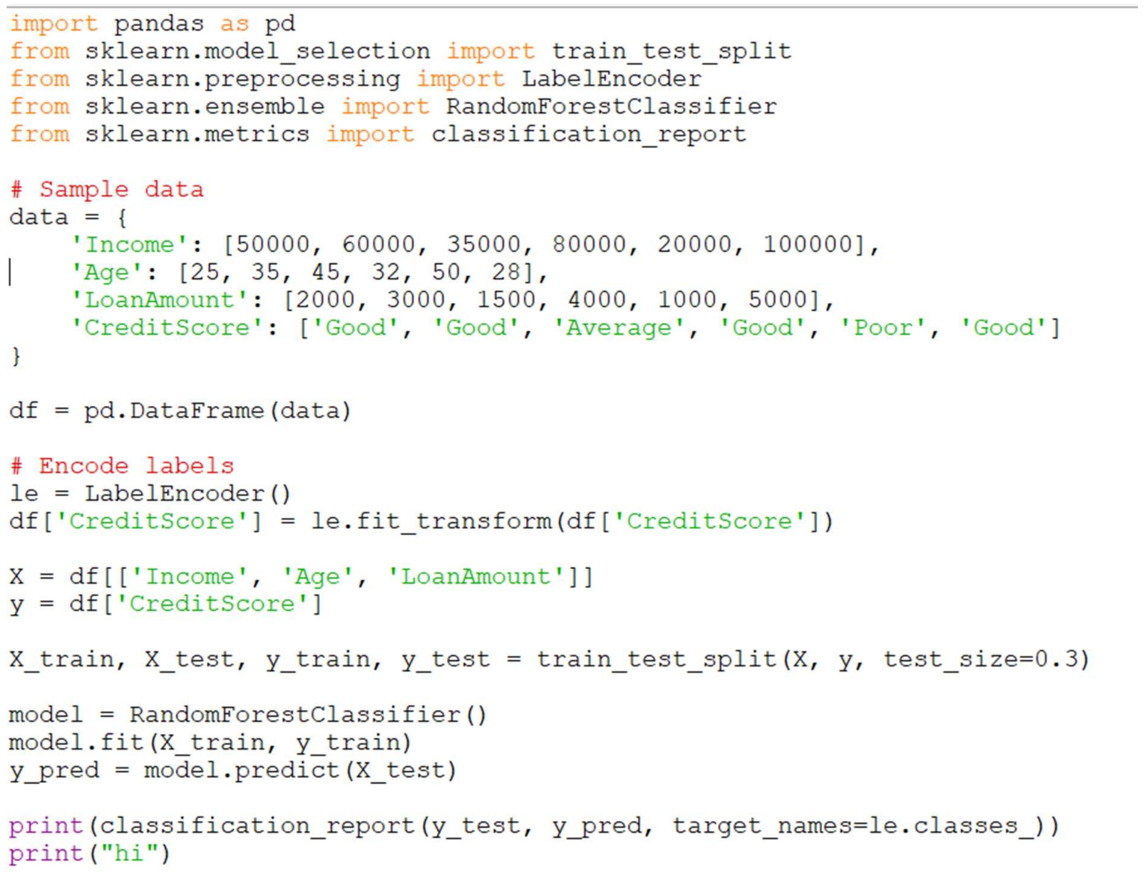
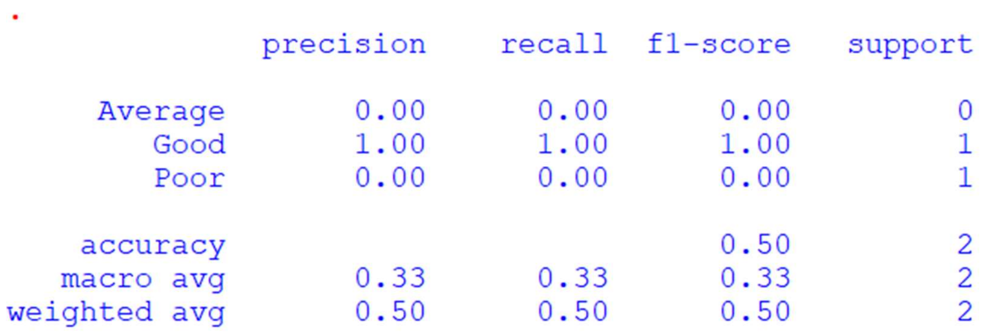
Experiment – 11



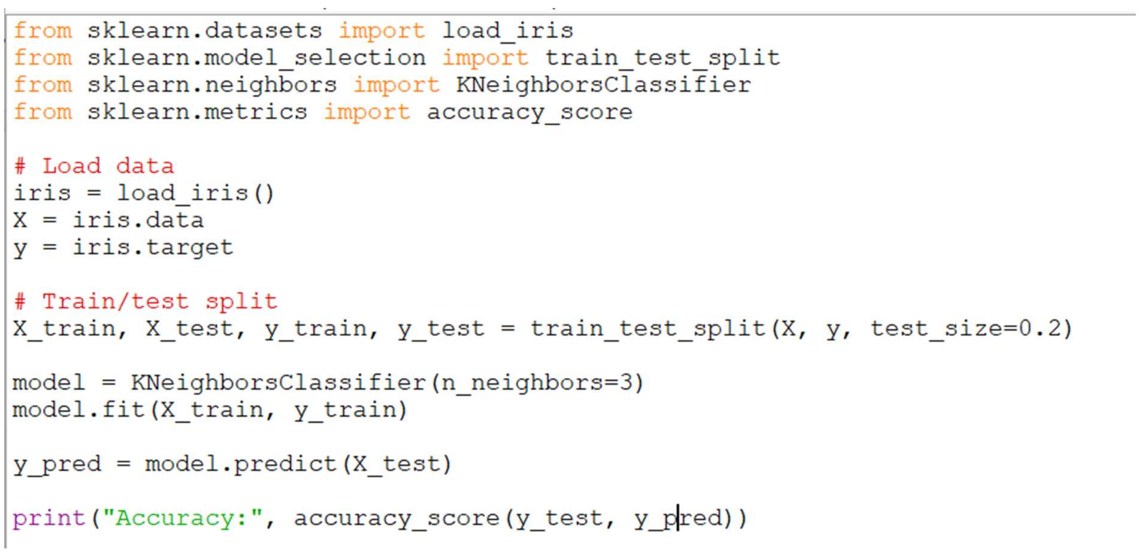
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Sample data
data = {
    'Income': [50000, 60000, 35000, 80000, 20000, 100000],
    'Age': [25, 35, 45, 32, 50, 28],
    'LoanAmount': [2000, 3000, 1500, 4000, 1000, 5000],
    'CreditScore': ['Good', 'Good', 'Average', 'Good', 'Poor', 'Good']
}

df = pd.DataFrame(data)

# Encode labels
le = LabelEncoder()
df['CreditScore'] = le.fit_transform(df['CreditScore'])

X = df[['Income', 'Age', 'LoanAmount']]
y = df['CreditScore']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred, target_names=le.classes_))
print("hi")
```

Output



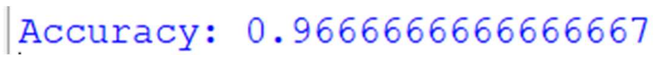
```
              precision    recall  f1-score   support

     Average       0.00      0.00      0.00         0
        Good       1.00      1.00      1.00         1
        Poor       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.33      0.33      0.33         2
weighted avg       0.50      0.50      0.50         2
```

Experiment – 12



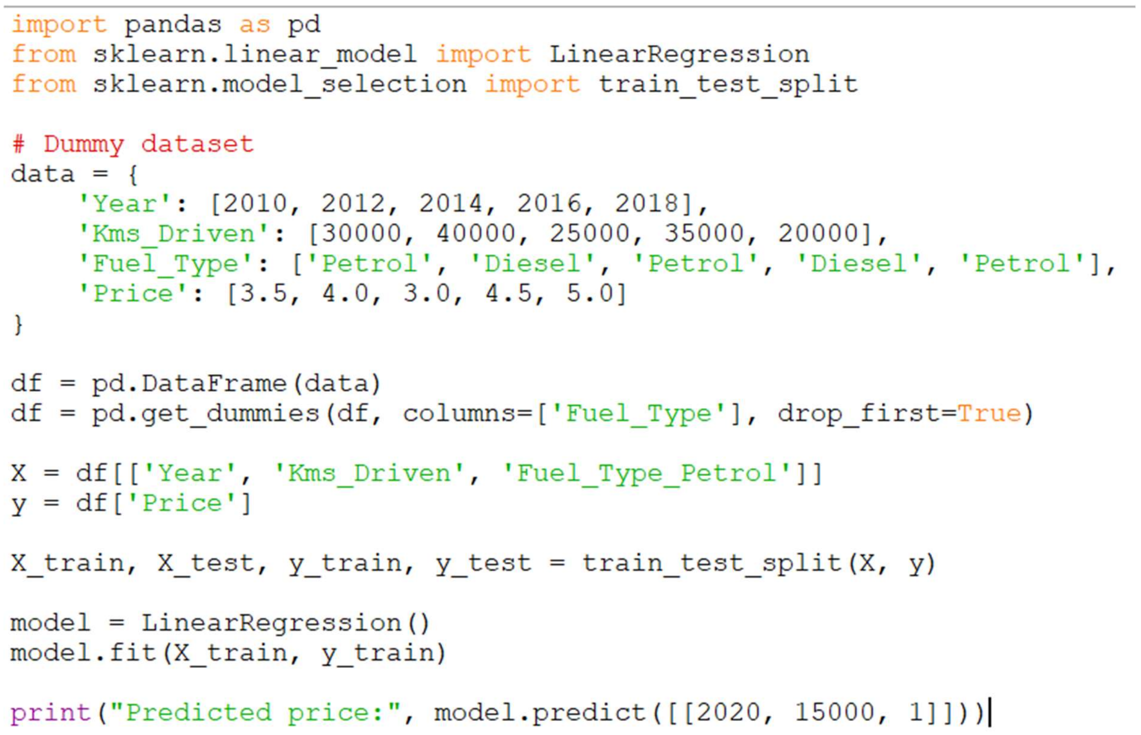
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load data
iris = load_iris()
X = iris.data
y = iris.target

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

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

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

Output



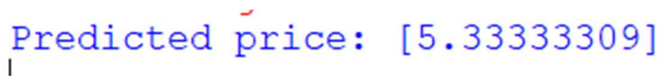
```
Accuracy: 0.9666666666666667
```

Experiment – 13



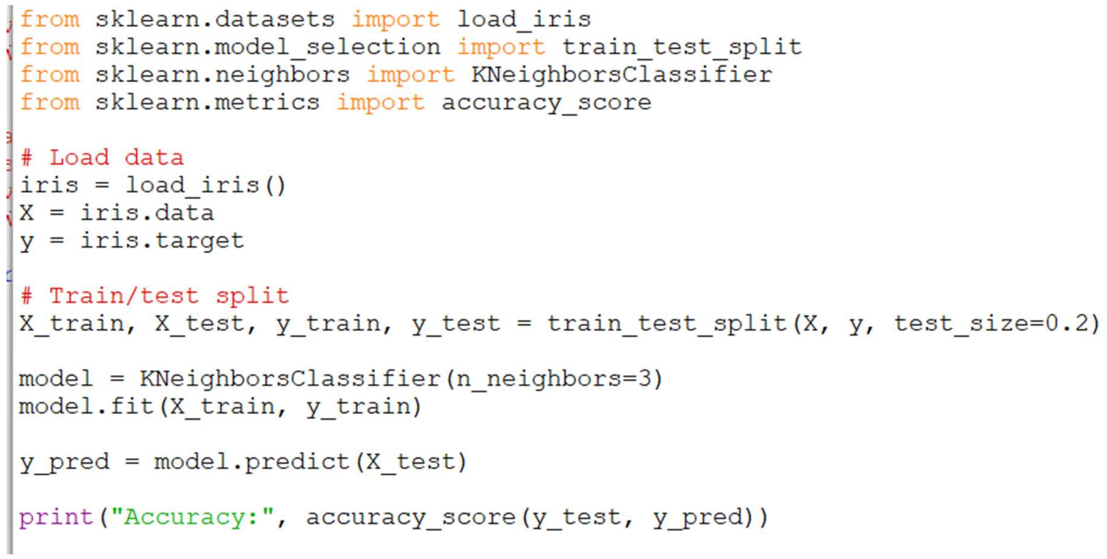
```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Dummy dataset
data = {
    'Year': [2010, 2012, 2014, 2016, 2018],
    'Kms_Driven': [30000, 40000, 25000, 35000, 20000],
    'Fuel_Type': ['Petrol', 'Diesel', 'Petrol', 'Diesel', 'Petrol'],
    'Price': [3.5, 4.0, 3.0, 4.5, 5.0]
}

df = pd.DataFrame(data)
df = pd.get_dummies(df, columns=['Fuel_Type'], drop_first=True)

X = df[['Year', 'Kms_Driven', 'Fuel_Type_Petrol']]
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y)

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

print("Predicted price:", model.predict([[2020, 15000, 1]]))
```

Output



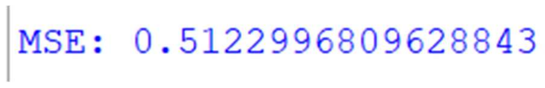
```
Predicted price:  [5.33333309]
```

Experiment – 14



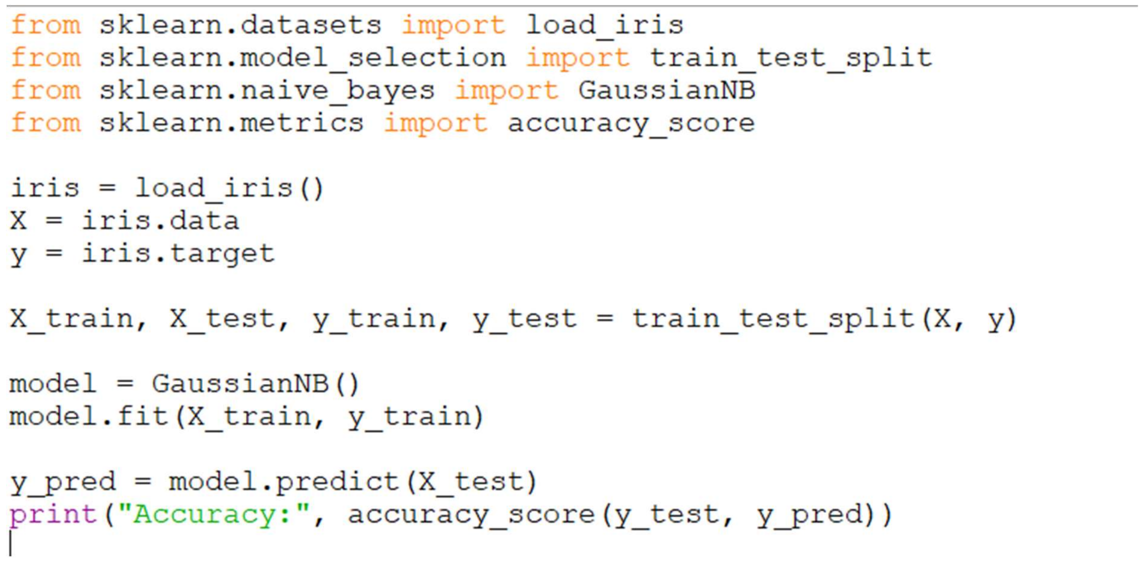
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load data
iris = load_iris()
X = iris.data
y = iris.target

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

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

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

Output



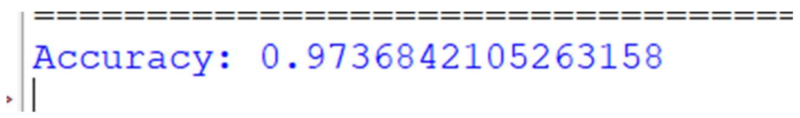
```
MSE: 0.5122996809628843
```

Experiment – 15



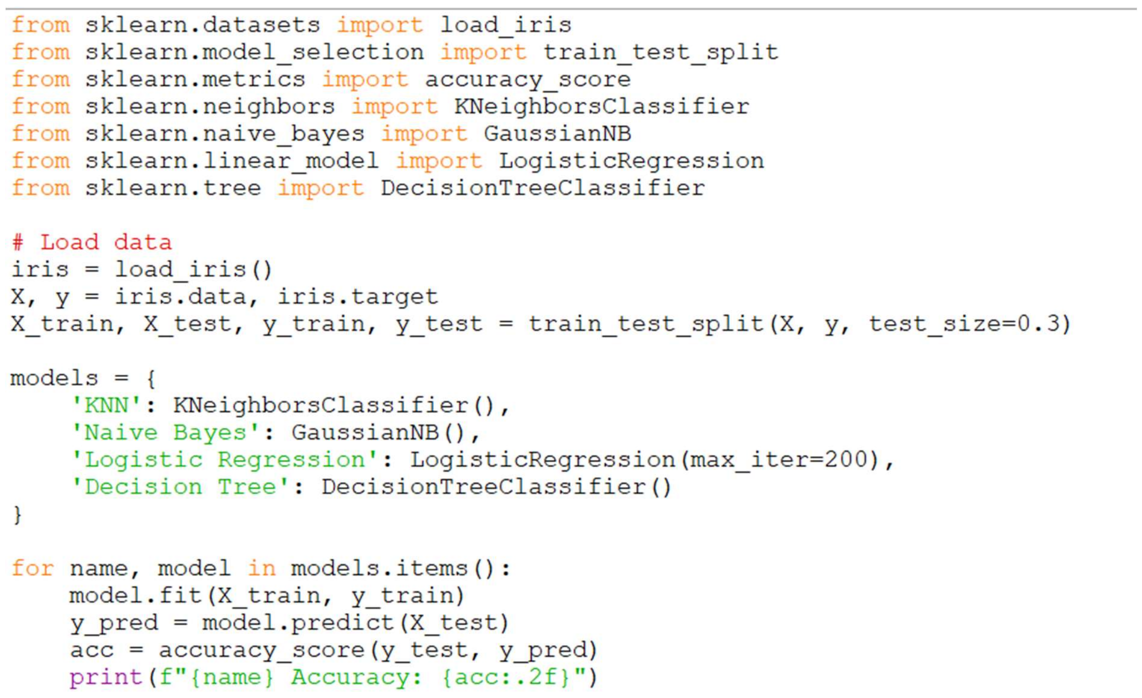
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y)

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

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Output



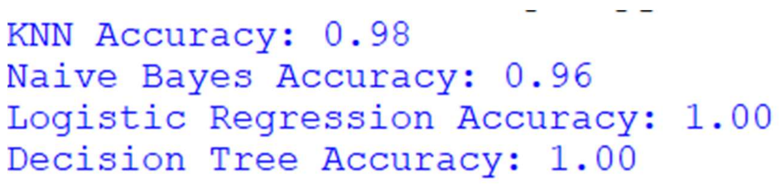
```
==================================
Accuracy: 0.9736842105263158
```

Experiment – 16



```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

# Load data
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

models = {
    'KNN': KNeighborsClassifier(),
    'Naive Bayes': GaussianNB(),
    'Logistic Regression': LogisticRegression(max_iter=200),
    'Decision Tree': DecisionTreeClassifier()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {acc:.2f}")
```

Output



```
KNN Accuracy: 0.98
Naive Bayes Accuracy: 0.96
Logistic Regression Accuracy: 1.00
Decision Tree Accuracy: 1.00
```

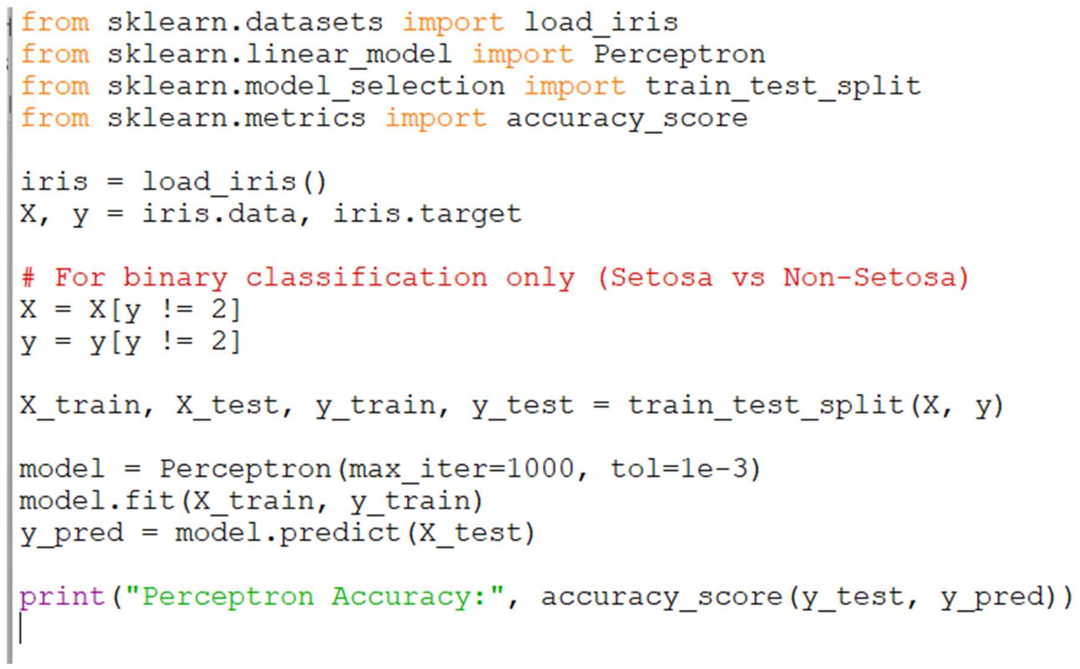
Experiment – 17

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Dummy data (create your own or replace with real dataset)
data = {
    'RAM': [4, 6, 8, 3, 12],
    'ROM': [64, 128, 256, 32, 512],
    'Battery': [4000, 5000, 4500, 3000, 6000],
    'Price': [15000, 20000, 25000, 10000, 40000]
}

df = pd.DataFrame(data)

X = df[['RAM', 'ROM', 'Battery']]
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y)

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

print("Predicted Price for 6GB RAM, 128GB ROM, 5000mAh battery:")
print(model.predict([[6, 128, 5000]]))
```

Output

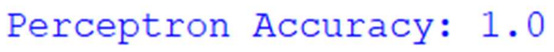
```
[21363.94487889]
```

Experiment – 18



```
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
X, y = iris.data, iris.target

# For binary classification only (Setosa vs Non-Setosa)
X = X[y != 2]
y = y[y != 2]

X_train, X_test, y_train, y_test = train_test_split(X, y)

model = Perceptron(max_iter=1000, tol=1e-3)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

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

Output



```
Perceptron Accuracy: 1.0
```

Experiment – 19

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

# Sample dummy data
data = {
    'Age': [25, 40, 35, 28, 50],
    'Income': [30000, 50000, 40000, 32000, 60000],
    'Credit_Score': ['Good', 'Poor', 'Average', 'Good', 'Poor'],
    'Loan_Approved': ['Yes', 'No', 'Yes', 'Yes', 'No']
}

df = pd.DataFrame(data)

le = LabelEncoder()
df['Credit_Score'] = le.fit_transform(df['Credit_Score'])
df['Loan_Approved'] = le.fit_transform(df['Loan_Approved'])

X = df[['Age', 'Income', 'Credit_Score']]
y = df['Loan_Approved']

X_train, X_test, y_train, y_test = train_test_split(X, y)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

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

Output

```
Accuracy: 0.0
```

Experiment – 20

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import numpy as np

# Dummy monthly sales data
data = {
    'Month': [1, 2, 3, 4, 5, 6],
    'Sales': [1000, 1500, 2000, 2500, 3000, 3500]
}

df = pd.DataFrame(data)

X = df[['Month']]
y = df['Sales']

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

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

# Predict future sales for month 7
future_month = np.array([[7]])
future_sales = model.predict(future_month)
print(f"Predicted Sales for Month 7: {future_sales[0]:.2f}")
```

Output

```
Predicted Sales for Month 7: 4000.00
```