statistics-lab-9-assignment

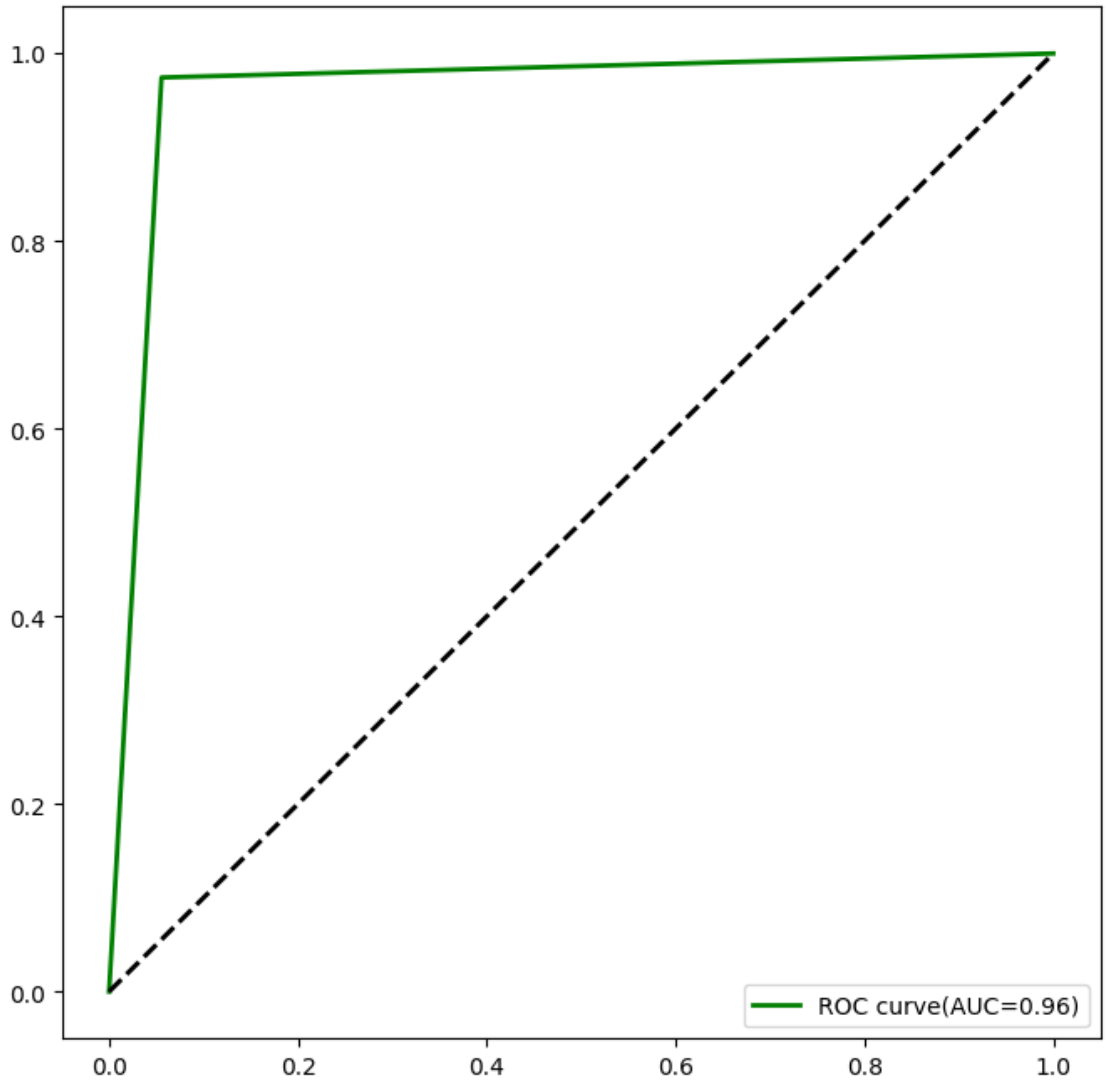
December 13, 2023

Q1.Demonstrate the usage of predict_proba function in Python.

[3]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,roc_curve,auc
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier

cancer = load_breast_cancer()

#Splitting data into independent and dependent variables
X = cancer.data
y = cancer.target

# Splitting data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2,␣
 ↪random_state=38)

# Making Classification Model
model = DecisionTreeClassifier(random_state=38)
model.fit(X_train, y_train)

# Probability Estimation
probabilities = model.predict_proba(X_test)
print("Predicted Prbabilities : ", probabilities[:5])
y1_prob = model.predict_proba(X_test)[:,1]

# ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y1_prob)

# AUC
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8,8))
plt.plot(fpr, tpr, color='green', lw=2, label=f'ROC curve(AUC={roc_auc:.2f})')
plt.plot([0,1],[0,1], color='black', lw=2, linestyle = '--')
plt.legend(loc='lower right')
```

```
plt.show()
```

```
Predicted Prbabilities :  [[1. 0.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [1. 0.]]
```



Q2.Explain about Monte-Carlo simulation. Explain with dice-game.

Ans= Monto-Carlo Simulation:: Monte Carlo simulation is a mathematical technique that estimates the possible outcomes of an uncertain event based on repeated random sampling. It uses a probability distribution to generate a range of results and calculate the probability of each result occurring. The Monte Carlo Method was invented by John von Neumann and Stanislaw Ulam

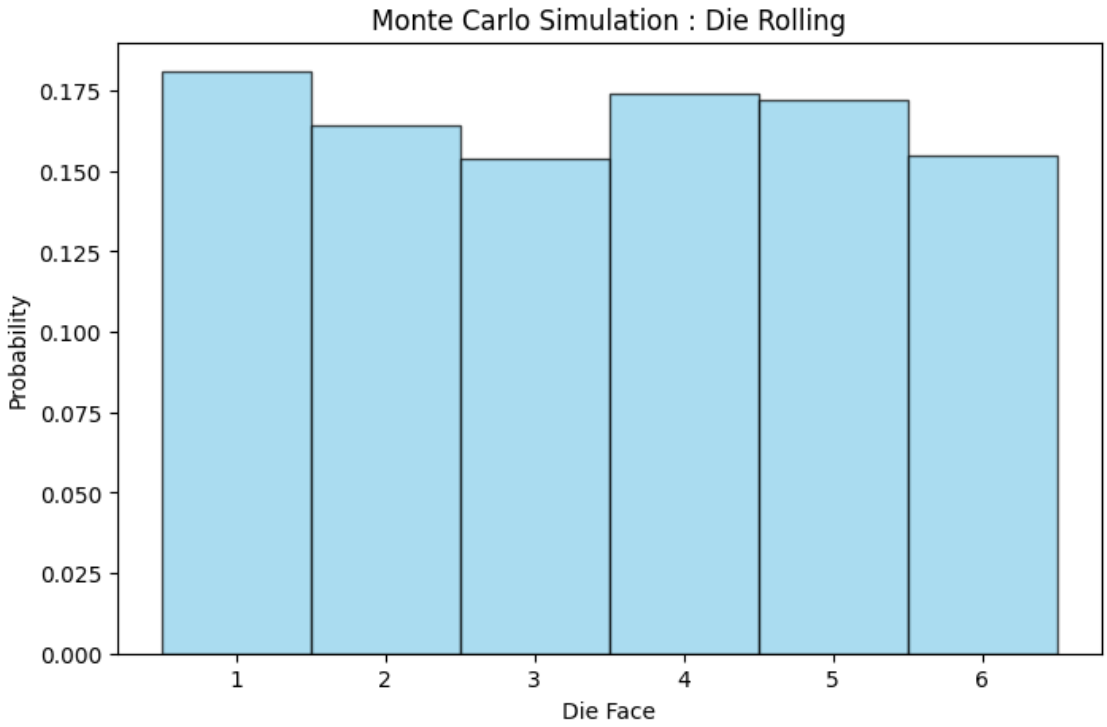
during World War II to improve decision making under uncertain conditions. It was named after a well-known casino town, called Monaco, since the element of chance is core to the modeling approach, similar to a game of roulette. Monte Carlo simulations have assessed the impact of risk in many real-life scenarios, such as in artificial intelligence, stock prices, sales forecasting, project management, and pricing.

The Monte Carlo simulation involves three basic steps:

1. Set up the predictive model, identifying both the dependent variable to be predicted nd the independent variables (also known as the input, risk or predictor variables) that will drive the prediction.
2. Define the range of values for each input variable.
3. Run the simulation multiple times, each time using a different set of random numbers be ween the minimum and maximum values. In a typical Monte Carlo experi ment, this exercise can be repeated thousands of times to produce a large number of likely outcomes.

Monte Carlo simulations are also utilized for long-term predictions due to their acc uracy. As the number of inputs increase, the number of forecasts also grows, aing you to roject outcomes farther out in time with more a.simulation.asp.

[6]:
```
import numpy as np
import matplotlib.pyplot as plt

num_simulations = 1000
np.random.seed(42)

simulate_outcomes = np.random.randint(1,7, size=(num_simulations,))
plt.figure(figsize=(8,5))
plt.hist(simulate_outcomes, bins=np.arange(0.5,7.5,1),density=True, alpha=0.7,␣
 ↪color='skyblue', edgecolor='black')
plt.title("Monte Carlo Simulation : Die Rolling")
plt.xlabel("Die Face")
plt.ylabel("Probability")
plt.xticks(np.arange(1,7,1))
plt.show()
```



Q3. Consider a chocolate manufacturing company that produces only two types of chocolates — A and B. Both the chocolates require Milk and Choco only. To manufacture each unit of A and B, the following quantities are required: Each unit of A requires 1 unit of Milk and 3 units of Choco Each unit of B requires 1 unit of Milk and 2 units of Choco The company kitchen has a total of 5 units of Milk and 12 units of Choco. On each sale, the company makes a profit of Rs 6 per unit A sold, Rs 5 per unit B sold. Now, the company wishes to maximize its profit. How many units of A and B should it produce, respectively? Model this problem by using Python and PuLP and solve it.

[7]:
```
!pip install pulp
```

```
Collecting pulp
  Downloading PuLP-2.7.0-py3-none-any.whl (14.3 MB)
                              14.3/14.3 MB
38.6 MB/s eta 0:00:00
Installing collected packages: pulp
Successfully installed pulp-2.7.0
```

[8]:
```
from pulp import *

# Create the 'prob' variable to contain the problem data
prob = LpProblem("Chocolate Manufacturing Problem", LpMaximize)
```

```
# Define decision variables
a = LpVariable("Units of A", 0, None, LpInteger)
b = LpVariable("Units of B", 0, None, LpInteger)

# Define objective function
prob += 6*a + 5*b, "Total Profit"

# Define constraints
prob += a + b <= 5, "Milk_Constraint"
prob += 3*a + 2*b <= 12, "Choco_Constraint"

# Solve the problem
prob.solve()

# Print the results
print("Number of units of A to produce: ", int(a.varValue))
print("Number of units of B to produce: ", int(b.varValue))
```

```
Number of units of A to produce:  2
Number of units of B to produce:  3
```

```
/usr/local/lib/python3.10/dist-packages/pulp/pulp.py:1352: UserWarning: Spaces
are not permitted in the name. Converted to '_'
  warnings.warn("Spaces are not permitted in the name. Converted to '_'")
```

Q4.A manufacturer manufactures chairs and tables. Manufacturer wants to increase the profit. 200 profit per chair and 300 profit per tables. A chair requires 1 hour of finishing labor and 2 hours of carpentery labor. A table requires 2 hours of finishing labor and 1 hour of carpentery labor. Each week the manufacturer has only 100 finishing hours and 100 carpentery hours available. Solve using python.

[9]:
```
from pulp import *

# Create the 'prob' variable to contain the problem data
prob = LpProblem("Maximizing Profit", LpMaximize)

# Define decision variables
x = LpVariable("Number of chairs", 0, None, LpInteger)
y = LpVariable("Number of tables", 0, None, LpInteger)

# Define objective function
prob += 200*x + 300*y, "Total Profit"

# Define constraints
prob += x + 2*y <= 100, "Finishing Hours Constraint"
prob += 2*x + y <= 100, "Carpentry Hours Constraint"

# Solve the problem
```

```
prob.solve()

# Print the results
print("Number of chairs to produce: ", int(x.varValue))
print("Number of tables to produce: ", int(y.varValue))
```

```
Number of chairs to produce:  32
Number of tables to produce:  34
```

Q5.Goal is to maximize sales of a phone company. It produces foldyphone and tinyphones. Store need 200 tinyphones for contracts, 500 foldyphones. Factory has 2999.5 hours of production time. Time consumed for foldyphone is 1.5 hours and tinyphone is 2 hours. How many of each phone should we make to maximize the sales. Solve using python.

[11]:
```
from pulp import *

# LP problem
problem = LpProblem("Maximize_Sales", LpMaximize)

# decision variables
foldyphones = LpVariable("Foldyphones", lowBound=0, cat="Integer")
tinyphones = LpVariable("Tinyphones", lowBound=0, cat="Integer")

# objective function
problem += foldyphones + tinyphones, "Total_Sales"

# constraints
problem += tinyphones >= 200, "Tinyphone_Constraint"
problem += foldyphones >= 500, "Foldyphone_Constraint"
problem += 1.5 * foldyphones + 2 * tinyphones <= 2999.5,␣
 ↪"Production_Time_Constrain"

# Solve the LP problem
problem.solve()

# Print the optimal solution
print("Optimal Solution:")
print("Foldyphones =", value(foldyphones))
print("Tinyphones =", value(tinyphones))
print("Total Sales =", value(problem.objective))
```

```
Optimal Solution:
Foldyphones = 1733.0
Tinyphones = 200.0
Total Sales = 1933.0
```