NAME : Mohammad Suhail **Course : BCA 6th Sem**

Roll no : 2121259 (57) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to use different techniques for filling the missing values.

SOURCE CODE :

```
import numpy as np
import pandas as pd
data = {
'A': [1, 2, np.nan, 4, 5],
'B': [np.nan, 6, 7, np.nan, 9],
'C': [10, 11, 12, np.nan, 14],
'D': [np.nan, np.nan, np.nan, np.nan, np.nan]
}
df = pd.DataFrame(data)
df_dropped = df.dropna()
df_filled = df.fillna(0)
df_mean_filled = df.fillna(df.mean())
df_median_filled = df.fillna(df.median())
df_ffilled = df.ffill()
df_bfilled = df.bfill()
print("Original DataFrame:")
print(df)
print("\nDataFrame after removing rows with missing values:")
print(df_dropped)
print("\nDataFrame after filling missing values with 0:")
print(df_filled)
print("\nDataFrame after filling missing values with column means:")
print(df_mean_filled)
print("\nDataFrame after filling missing values with column medians:")
print(df_median_filled)
print("\nDataFrame after forward filling missing values:")
print(df_ffilled)
print("\nDataFrame after backward filling missing values:")
print(df_bfilled)
```

NAME : Mohammad Suhail **Course : BCA 6th Sem**

Roll no : 2121259 (57) **Section : C**

Subject : Fundamentals of Machine Learning

OUTPUT :

```
Original DataFrame:
       A      B       C     D
0 1.0 NaN 10.0 NaN
1  2.0   6.0   11.0 NaN
2  NaN   7.0   12.0 NaN
3   4.0   NaN     NaN NaN
4 5.0 9.0 14.0 NaN
DataFrame after removing rows with missing values:
Empty DataFrame
Columns: [A, B, C, D]
Index: []
DataFrame after filling missing values with 0:
       A      B       C      D
0   1.0   0.0   10.0   0.0
1   2.0   6.0   11.0   0.0
2   0.0   7.0   12.0   0.0
3   4.0   0.0    0.0   0.0
4   5.0   9.0   14.0   0.0
DataFrame after filling missing values with column means:
       A             B         C     D
0   1.0   7.333333   10.00 NaN
1   2.0   6.000000   11.00 NaN
2   3.0   7.000000   12.00 NaN
3   4.0   7.333333   11.75 NaN
4   5.0   9.000000   14.00 NaN
DataFrame after filling missing values with column medians:
       A      B       C     D
0   1.0   7.0   10.0 NaN
1   2.0   6.0   11.0 NaN
2   3.0   7.0   12.0 NaN
3   4.0   7.0   11.5 NaN
4   5.0   9.0   14.0 NaN
DataFrame after forward filling missing values:
       A      B       C     D
0  1.0    NaN   10.0 NaN
1   2.0   6.0   11.0 NaN
2   2.0   7.0   12.0 NaN
3   4.0   7.0   12.0 NaN
4   5.0   9.0   14.0 NaN
DataFrame after backward filling missing values:
       A      B       C     D
0   1.0   6.0   10.0 NaN
1   2.0   6.0   11.0 NaN
2   4.0   7.0   12.0 NaN
3   4.0   9.0   14.0 NaN
4   5.0   9.0   14.0 NaN
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to generate the synthetic dataset.

SOURCE CODE :

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
california_housing=fetch_california_housing()
x=california_housing.data
feature_names=california_housing.feature_names
df=pd.DataFrame(data=x,columns=feature_names)
nan_indices=np.random.choice(df.index,size=int(0.1*df.size),replace=False)
rows,cols=np.unravel_index(nan_indices,df.shape)
df.values[rows,cols]=np.nan
missing_values_per_column=df.isnull().sum()
print("missing values in each column:")
print(missing_values_per_column)
```

OUTPUT :

```
missing values in each column:
MedInc          2054
HouseAge         2073
AveRooms         2086
AveBedrms        2076
Population      2048
AveOccup         2095
Latitude        2058
Longitude       2022
dtype: int64
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to add two matrices

SOURCE CODE :

```
A= []
print("Enter 9 Elements for First Matrix: ")
for i in range(3):
    A.append([])
    for j in range(3):
        num = int(input())
        A[i].append(num)
B = []
print("Enter 9 Elements for Second Matrix: ")
for i in range(3):
    B.append([])
    for j in range(3):
        num = int(input())
        B[i].append(num)
Result = []
for i in range(3):
    Result.append([])
    for j in range(3):
        Result[i].append(A[i][j]+B[i][j])
print("\nAddition Result of Two Given Matrix is:")
for i in range(3):
    for j in range(3):
        print(Result[i][j], end=" ")
    print()
```

OUTPUT :

```
Enter 9 Elements for First Matrix:
5
3
7
2
8
2
9
5
3
Enter 9 Elements for Second Matrix:
2
8
2
9
4
1
3
4
5
Addition Result of Two Given Matrix is:
7 11 9
11 12 3
12 9 8 4
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to multiply matrices

SOURCE CODE :

```
def multiply_matrices():
    m1 = int(input("Enter the number of rows for matrix1: "))
    n1 = int(input("Enter the number of columns for matrix1: "))
    m2 = int(input("Enter the number of rows for matrix2: "))
    n2 = int(input("Enter the number of columns for matrix2: "))
    if n1 != m2:
        print("Cannot multiply matrices. Number of columns in matrix1 should be equal to the number
of rows in matrix2.")
        return
    matrix1 = []
    matrix2 = []
    print("Enter elements of matrix1:")
    for i in range(m1):
        row = []
        for j in range(n1):
            row.append(int(input(f"Enter element ({i+1},{j+1}): ")))
        matrix1.append(row)
    print("Enter elements of matrix2:")
    for i in range(m2):
        row = []
        for j in range(n2):
            row.append(int(input(f"Enter element ({i+1},{j+1}): ")))
        matrix2.append(row)
    result = [[sum(matrix1[i][k] * matrix2[k][j] for k in range(n1)) for j in range(n2)] for i in range(m1)]
    print("Result:")
    for row in result:
        print(row)
multiply_matrices()
```

OUTPUT :

```
Enter the number of rows for matrix1: 3
Enter the number of columns for matrix1: 3
Enter the number of rows for matrix2: 3
Enter the number of columns for matrix2: 3
Enter elements of matrix1:
Enter element (1,1): 6
Enter element (1,2): 4
Enter element (1,3): 7
Enter element (2,1): 3
Enter element (2,2): 7
Enter element (2,3): 8
Enter element (3,1): 4
Enter element (3,2): 1
Enter element (3,3): 3
Enter elements of matrix2:
Enter element (1,1): 9
Enter element (1,2): 4
Enter element (1,3): 2
Enter element (2,1): 5
Enter element (2,2): 7
Enter element (2,3): 2
Enter element (3,1): 7
Enter element (3,2): 6
Enter element (3,3): 4
Result:
[123, 94, 48]
[118, 109, 52]
[62, 41, 22]
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to find the combinatorics of a given input number

SOURCE CODE :

```
def factorial(n):
      if(n==1 or n==0):
            return n
      else:
            return (n*factorial(n-1))
def permutations(n,r):
      return factorial(n)//factorial(n-r)
def combinations(n,r):
      return permutations(n,r)//factorial(r)
n=int(input("Enter the number of items :"))
r=int(input("Enter the number of items to be choosed :"))
print("The Combinations is : ",combinations(n,r))
```

OUTPUT :

```
Enter the number of items :5
Enter the number of items to be choosed :4
The Combinations is :     5
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT :Write a python program to find the permutation of a given number

SOURCE CODE :

```
def factorial(n):
    if(n==1 or n==0):
        return n
    else:
        return (n*factorial(n-1))
def permutations(n,r):
    return factorial(n)//factorial(n-r)
n=int(input("Enter the number of items :"))
r=int(input("Enter the number of items to be choosed :"))
print("The permutations ",permutations(n,r))
```

OUTPUT :

```
Enter the number of items :5
Enter the number of items to be choosed :3
The permutations 60
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to find factorial of a given number

SOURCE CODE :

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
num = int(input("Enter a number: "))
print("Factorial:", factorial(num))
factorial(n)
```

OUTPUT :

```
Enter a number: 5
Factorial: 120
120
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to print the transpose of a matrix
SOURCE CODE :

```
matrix = []
row = int(input("Enter number of rows: "))
col = int(input("Enter number of columns: "))
for i in range(row):
   for j in range(col):
      matrix.append([])
      num = int(input("Enter the element: "))
      matrix[i].append(num)
print("Input matrix: ")
for i in range (row):
   for j in range (col):
      print(matrix[i][j], end = " ")
   print()
transpose = []
for j in range(col):
   transpose.append([])
   for i in range (row):
      t_num = matrix[i][j]
      transpose[j].append(t_num)
print('Transpose matrix: ')
for i in range (row):
   for j in range (col):
      print (transpose[i][j], end = ' ')
   print()
```

OUTPUT :

```
Enter number of rows: 3
Enter number of columns: 3
Enter the element: 1
Enter the element: 3
Enter the element: 5
Enter the element: 7
Enter the element: 9
Enter the element: 2
Enter the element: 4
Enter the element: 6
Enter the element: 8
Input matrix:
1 3 5
7 9 2
4 6 8
Transpose matrix:
1 7 4
3 9 6
5 2 8
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to generate a dataset and implement a K-Means clustering algorithm.

SOURCE CODE :

```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
np.random.seed(42)
def PointsInCircum(r,n=100):
    return
[(math.cos(2*math.pi/n*x)*r+np.random.normal(0,30),math.sin(2*math.pi/n*x)*r+np.random.normal
(0,30)) for x in range(1,n+1)]
df=pd.DataFrame(PointsInCircum(500,1000))
df=df.append(PointsInCircum(300,700))
df=df.append(PointsInCircum(100,300))
df=df.append(pd.DataFrame([(np.random.randint(-600,600),np.random.randint(-600,600)) for i in
range(300)]))
print(df.head())
```

OUTPUT :

```
            0          1
0  514.891555  -1.006357
1  519.391178  51.973916
2  492.886575   2.400111
3  547.218479  35.588090
4  485.669049  31.982181
```

NAME : Mayank kumar Bamaniya **Course : BCA 6th Sem**

Roll no : 2121256 (55) **Section : C**

Subject : Fundamentals of Machine Learning

PROBLEM STATEMENT : Write a python program to data analyse using supervised algorithms building a predictive model for customer churn in a subscription-based business

SOURCE CODE :

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib
def generate_customer_churn_data(num_customers=1000, start_date='2019-01-01',
end_date='2022-01-01'):
    start_date = pd.to_datetime(start_date)
    end_date = pd.to_datetime(end_date)
    customer_ids = np.arange(1, num_customers + 1)
    join_dates = [np.random.choice(pd.date_range(start_date, end_date)) for _ in
range(num_customers)]
    churn_dates = [join_date + pd.Timedelta(days=np.random.randint(30, 365)) for join_date in
join_dates]
    churn_status = ['Churned' if date <= end_date else 'Active' for date in churn_dates]
    data = {
        'CustomerID': customer_ids,
        'JoinDate': join_dates,
        'ChurnDate': churn_dates,
        'ChurnStatus': churn_status
    }
    df = pd.DataFrame(data)
    return df
def preprocess_data(df):
    df['JoinYear'] = df['JoinDate'].dt.year
    df['JoinMonth'] = df['JoinDate'].dt.month
    df['JoinDay'] = df['JoinDate'].dt.day
    df['JoinDayOfWeek'] = df['JoinDate'].dt.dayofweek
    df['DaysToChurn'] = (df['ChurnDate'] - df['JoinDate']).dt.days
    df.drop(['JoinDate', 'ChurnDate'], axis=1, inplace=True)
    df['ChurnStatus'] = df['ChurnStatus'].map({'Active': 0, 'Churned': 1})
       return df
def split_data(df, test_size=0.2):
    X = df.drop('ChurnStatus', axis=1)
    y = df['ChurnStatus']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42)
    return X_train, X_test, y_train, y_test
def train_model(X_train, y_train):
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    return model
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
def save_model(model, filepath='customer_churn_model.pkl'):
    joblib.dump(model, filepath)
    print("Model saved successfully.")
def main():
    df = generate_customer_churn_data()
    df = preprocess_data(df)
    X_train, X_test, y_train, y_test = split_data(df)
    model = train_model(X_train, y_train)
    evaluate_model(model, X_test, y_test)
    save_model(model)
if _name_ == "_main_":
    main()
```

OUTPUT :

```
Accuracy: 0.97
Classification Report:
              precision    recall  f1-score   support
           0       0.94      0.88      0.91        34
           1       0.98      0.99      0.98       166
    accuracy                           0.97       200
   macro avg       0.96      0.94      0.95       200
weighted avg       0.97      0.97      0.97       200
Confusion Matrix:
[[ 30   4]
 [  2 164]]
Model saved successfully.
```