Pandas, Numpy, Matplotlib, seaborn and more

In [39]:

```
# Importing necessary libraries for data analysis
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical computations
import matplotlib.pyplot as plt  # For creating static, animated, and int
eractive visualizations
import seaborn as sns  # For statistical data visualization based on Matp
lotlib
import scipy  # For scientific and technical computing (including optimiz
ation, integration, and statistics)
from sklearn.preprocessing import StandardScaler, LabelEncoder  # For pre
processing data (scaling, encoding)
from sklearn.model_selection import train_test_split  # For splitting dat
a into training and testing sets
from sklearn.linear_model import LinearRegression  # For linear regressio
n models
from sklearn.metrics import mean_squared_error, r2_score  # For model eva
luation metrics
import statsmodels.api as sm  # For statistical modeling and hypothesis t
esting
```

In [40]:

```
dict1 = {
    "name":['satender', 'rohan', 'rahul', 'ajay'],
    "marks":[92, 34, 34, 17],
    "city":['london', 'ontario', 'ajex', 'windsor']
}
```

In [41]:

```
df = pd.DataFrame(dict1)
```

In [42]:

```
df
```

Out[42]:

	name	marks	city
0	satender	92	london
1	rohan	34	ontario
2	rahul	34	ajex
3	ajay	17	windsor

In [43]:

```
import pandas as pd
import numpy as np
```

```
dict1 = {
    "name": ['satender', 'rohan', 'rahul', 'ajay'],
    "marks": [92, 34, 34, 17],
    "city": ['london', 'ontario', 'ajex', 'windsor']
}
```

In [44]:

```
df = pd.DataFrame(dict1)
```

In [45]:

```
df
```

Out[45]:

	name	marks	city
0	satender	92	london
1	rohan	34	ontario
2	rahul	34	ajex
3	ajay	17	windsor

In [46]:

```
import pandas as pd
import numpy as np

# Define the DataFrame
dict1 = {
    "name": ['satender', 'rohan', 'rahul', 'ajay'],
    "marks": [92, 34, 34, 17],
    "city": ['london', 'ontario', 'ajex', 'windsor']
}

df = pd.DataFrame(dict1)

# Add a calculated column 'grade' based on marks
df['grade'] = np.select(
    [
        df['marks'] >= 90,
        df['marks'] >= 75,
        df['marks'] >= 50,
        df['marks'] >= 35
    ],
    ['A', 'B', 'C', 'D'],
    default='F'
)
```

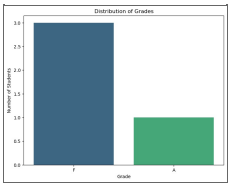
In [47]:

```
print(df)
```

```
       name  marks     city grade
  0  satender     92   london     A
  1     rohan     34  ontario     F
  2     rahul     34     ajex     F
  3      ajay     17  windsor     F
```

In [48]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Count the number of students in each grade
grade_counts = df['grade'].value_counts()

# Create a bar plot with the updated syntax
plt.figure(figsize=(8, 6))
sns.barplot(x=grade_counts.index, y=grade_counts.values, hue=grade_counts
.index, dodge=False, palette='viridis', legend=False)
plt.title('Distribution of Grades')
plt.xlabel('Grade')
plt.ylabel('Number of Students')
plt.show()
```



In [49]:

```
import pandas as pd
import numpy as np

# Original DataFrame
dict1 = {
    "name": ['satender', 'rohan', 'rahul', 'ajay'],
    "marks": [92, 34, 34, 17],
    "city": ['london', 'ontario', 'ajex', 'windsor']
}
df = pd.DataFrame(dict1)

# Add a column for pass/fail status based on marks
df['status'] = df['marks'].apply(lambda x: 'Pass' if x >= 35 else 'Fail')

# Add a column for a custom grade based on multiple criteria
def custom_grade(row):
    if row['marks'] >= 90:
        return 'A+'
    elif row['marks'] >= 75:
        return 'A'
    elif row['marks'] >= 50:
        return 'B'
    elif row['marks'] >= 35:
        return 'C'
    else:
        return 'F'

df['custom_grade'] = df.apply(custom_grade, axis=1)

print(df)
```

```
      name  marks     city status custom_grade
0  satender     92   london   Pass           A+
1     rohan     34  ontario   Fail            F
2     rahul     34     ajex   Fail            F
3      ajay     17  windsor   Fail            F
```

In [50]:

```
# Group by city and calculate the mean, sum, and count of marks
city_group = df.groupby('city')['marks'].agg(['mean', 'sum', 'count'])

# Add a column for the rank of each student within their city
df['rank_within_city'] = df.groupby('city')['marks'].rank(ascending=False
)

print(city_group)
print(df)
```

```
          mean  sum  count
city
ajex      34.0   34      1
london    92.0   92      1
ontario   34.0   34      1
windsor   17.0   17      1
       name  marks     city status custom_grade  rank_within_cit
y
0  satender     92   london   Pass           A+               1.
0
1     rohan     34  ontario   Fail            F               1.
0
2     rahul     34     ajex   Fail            F               1.
0
3      ajay     17  windsor   Fail            F               1.
0
```

In [51]:

```
# Create a pivot table to show the average marks and count of students by
 city and status
pivot_table = pd.pivot_table(df, values='marks', index='city', columns='s
tatus', aggfunc={'marks': ['mean', 'count']})

print(pivot_table)
```

```
         count       mean
status    Fail Pass  Fail  Pass
city
ajex       1.0  NaN  34.0   NaN
london     NaN  1.0   NaN  92.0
ontario    1.0  NaN  34.0   NaN
windsor    1.0  NaN  17.0   NaN
```

In [52]:

```
# Additional DataFrame with more student information
additional_info = pd.DataFrame({
    'name': ['satender', 'rohan', 'rahul', 'ajay', 'vikram'],
    'age': [22, 21, 22, 23, 24],
    'gender': ['M', 'M', 'M', 'M', 'M']
})

# Merge on the 'name' column
merged_df = pd.merge(df, additional_info, on='name', how='left')

print(merged_df)
```

```
        name  marks     city status custom_grade  rank_within_cit
y  age gender
0  satender     92   london   Pass           A+               1.
0   22      M
1     rohan     34  ontario   Fail            F               1.
0   21      M
2     rahul     34     ajex   Fail            F               1.
0   22      M
3      ajay     17  windsor   Fail            F               1.
0   23      M
```

In [53]:

```
# Introduce some missing data for demonstration
merged_df.loc[merged_df['name'] == 'vikram', 'marks'] = np.nan

# Fill missing data with the mean of the column
merged_df['marks_filled'] = merged_df['marks'].fillna(merged_df['marks'].
mean())

# Alternatively, interpolate missing values based on surrounding data
merged_df['marks_interpolated'] = merged_df['marks'].interpolate()

print(merged_df)
```

```
        name  marks     city status custom_grade  rank_within_cit
y  age gender  \
0  satender   92.0   london   Pass           A+               1.
0   22      M
1     rohan   34.0  ontario   Fail            F               1.
0   21      M
2     rahul   34.0     ajex   Fail            F               1.
0   22      M
3      ajay   17.0  windsor   Fail            F               1.
0   23      M

   marks_filled  marks_interpolated
0          92.0                92.0
1          34.0                34.0
2          34.0                34.0
3          17.0                17.0
```

In [54]:

```
# Filter students who passed and scored more than the average marks of th
e class
average_marks = df['marks'].mean()
filtered_students = df[(df['status'] == 'Pass') & (df['marks'] > average_
marks)]

print(filtered_students)
```

```
        name  marks    city status custom_grade  rank_within_city
0  satender     92  london   Pass           A+               1.0
```

In [55]:

```
# Calculate a rolling average of marks with a window size of 2
df['rolling_avg'] = df['marks'].rolling(window=2).mean()

print(df)
```

```
      name  marks     city status custom_grade  rank_within_cit
y  rolling_avg
0  satender     92   london   Pass           A+               1.
0         NaN
1     rohan     34  ontario   Fail            F               1.
0        63.0
2     rahul     34     ajex   Fail            F               1.
0        34.0
3      ajay     17  windsor   Fail            F               1.
0        25.5
```

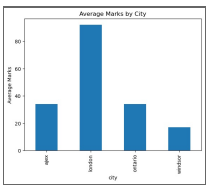
In [56]:

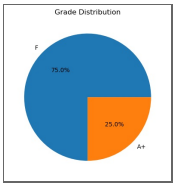
```
# Custom function to generate a summary for each row
def summarize_row(row):
    return f"{row['name']} from {row['city']} scored {row['marks']} and g
ot grade {row['custom_grade']}."

df['summary'] = df.apply(summarize_row, axis=1)

print(df['summary'])
```

```
0    satender from london scored 92 and got grade A+.
1       rohan from ontario scored 34 and got grade F.
2          rahul from ajex scored 34 and got grade F.
3        ajay from windsor scored 17 and got grade F.
Name: summary, dtype: object
```

In [57]:

```
import matplotlib.pyplot as plt

# Bar plot for average marks by city
df.groupby('city')['marks'].mean().plot(kind='bar', title='Average Marks
by City')
plt.ylabel('Average Marks')
plt.show()

# Pie chart for distribution of grades
df['custom_grade'].value_counts().plot(kind='pie', title='Grade Distribut
ion', autopct='%1.1f%%')
plt.ylabel('')
plt.show()
```





In [58]:

```
# Set a MultiIndex with 'city' and 'name'
df_multiindex = df.set_index(['city', 'name'])
```

```
# Access data using the MultiIndex
print(df_multiindex.loc['london'].loc['satender'])

# Reset the index to default
df_reset = df_multiindex.reset_index()
print(df_reset)
```

```
marks
92
status
Pass
custom_grade
A+
rank_within_city
1.0
rolling_avg
NaN
summary             satender from london scored 92 and got grade
A+.
Name: satender, dtype: object
      city      name  marks status custom_grade  rank_within_cit
y  \
0   london  satender     92   Pass           A+               1.
0
1  ontario     rohan     34   Fail            F               1.
0
2     ajex     rahul     34   Fail            F               1.
0
3  windsor      ajay     17   Fail            F               1.
0

   rolling_avg                                              summary
0          NaN  satender from london scored 92 and got grade A+.
1         63.0     rohan from ontario scored 34 and got grade F.
2         34.0        rahul from ajex scored 34 and got grade F.
3         25.5      ajay from windsor scored 17 and got grade F.
```

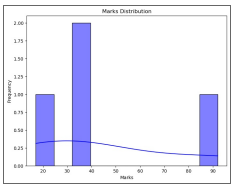
In [59]:

```
# Convert 'custom_grade' to a categorical type with order
grades = ['F', 'D', 'C', 'B', 'A', 'A+']
df['custom_grade'] = pd.Categorical(df['custom_grade'], categories=grades
, ordered=True)

# Sort DataFrame by categorical grade
df_sorted_by_grade = df.sort_values('custom_grade')

print(df_sorted_by_grade)
```

```
        name  marks     city status custom_grade  rank_within_cit
y  \
1     rohan     34  ontario   Fail            F               1.
0
2     rahul     34     ajex   Fail            F               1.
0
3      ajay     17  windsor   Fail            F               1.
0
0  satender     92   london   Pass           A+               1.
0

   rolling_avg                                              summary
1         63.0     rohan from ontario scored 34 and got grade F.
2         34.0        rahul from ajex scored 34 and got grade F.
3         25.5      ajay from windsor scored 17 and got grade F.
0          NaN  satender from london scored 92 and got grade A+.
```

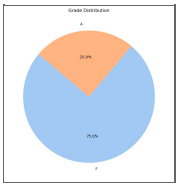
In [60]:

```
plt.figure(figsize=(8, 6))
sns.histplot(df['marks'], bins=10, kde=True, color='blue')
plt.title('Marks Distribution')
plt.xlabel('Marks')
plt.ylabel('Frequency')
plt.show()
```



In [61]:

```
plt.figure(figsize=(8, 8))
plt.pie(grade_counts, labels=grade_counts.index, autopct='%1.1f%%', start
angle=140, colors=sns.color_palette('pastel'))
plt.title('Grade Distribution')
plt.show()
```



In [66]:

```
# Binning an array of data
data = np.random.randn(1000)  # Generate 1000 random numbers
bins = np.linspace(-3, 3, 7)  # Create 6 bins between -3 and 3
digitized = np.digitize(data, bins)

# Counting the number of elements in each bin
bin_counts = np.bincount(digitized)
print("Bin Counts:", bin_counts)
```

```
Bin Counts: [  1  26 135 334 346 131  26   1]
```

In [67]:

```
import numpy as np

# Vectorized operation using ufunc
arr = np.array([1, 2, 3, 4, 5])

# Example of vectorized operation
result = np.exp(arr)  # Exponential function applied element-wise
print("Exponential of Array:", result)

# Custom ufunc
def custom_func(x, y):
    return x + y * y

# Vectorized using np.vectorize
vectorized_func = np.vectorize(custom_func)
result = vectorized_func(arr, arr)
print("Custom Ufunc Result:", result)
```

```
Exponential of Array: [  2.71828183   7.3890561   20.08553692  5
4.59815003 148.4131591 ]
Custom Ufunc Result: [ 2  6 12 20 30]
```

In [68]:

```
# Example of broadcasting without memory duplication
arr1 = np.array([[1], [2], [3]])
arr2 = np.array([4, 5, 6])

# Broadcasting adds arr2 to each row of arr1 without copying data
result = arr1 + arr2
print("Broadcasted Result:\n", result)

# Memory-efficient operation
print("Memory Address of arr1:", arr1.__array_interface__['data'])
print("Memory Address of arr2:", arr2.__array_interface__['data'])
```

```
Broadcasted Result:
 [[5 6 7]
 [6 7 8]
 [7 8 9]]
Memory Address of arr1: (2162781248528, False)
Memory Address of arr2: (2162781248048, False)
```

In [69]:

```
# Integer array indexing
arr = np.array([[1, 2], [3, 4], [5, 6]])
print("Original Array:\n", arr)

# Select elements using an integer array
result = arr[[0, 1, 2], [1, 0, 1]]  # Selects (0,1), (1,0), and (2,1)
print("Integer Array Indexing Result:", result)

# Boolean indexing
mask = arr > 2
print("Boolean Mask:\n", mask)
result = arr[mask]
print("Boolean Indexing Result:", result)
```

```
Original Array:
 [[1 2]
 [3 4]
 [5 6]]
Integer Array Indexing Result: [2 3 6]
Boolean Mask:
 [[False False]
 [ True  True]
 [ True  True]]
Boolean Indexing Result: [3 4 5 6]
```

In [70]:

```python
# Create a structured array with multiple data types
data = np.array([
    ('Satender', 92, 'London'),
    ('Rohan', 34, 'Ontario'),
    ('Rahul', 34, 'Ajex'),
    ('Ajay', 17, 'Windsor')
], dtype=[('name', 'U10'), ('marks', 'i4'), ('city', 'U10')])

print("Structured Array:\n", data)

# Accessing specific fields
print("Names:", data['name'])
print("Marks:", data['marks'])
```

```
Structured Array:
 [('Satender', 92, 'London') ('Rohan', 34, 'Ontario') ('Rahul',
34, 'Ajex')
 ('Ajay', 17, 'Windsor')]
Names: ['Satender' 'Rohan' 'Rahul' 'Ajay']
Marks: [92 34 34 17]
```

In [71]:

```python
# Create a masked array
arr = np.array([1, 2, 3, -1, 5])
masked_arr = np.ma.masked_array(arr, mask=[0, 0, 0, 1, 0])

print("Masked Array:", masked_arr)
print("Mean ignoring masked elements:", np.ma.mean(masked_arr))
```

```
Masked Array: [1 2 3 -- 5]
Mean ignoring masked elements: 2.75
```

In [72]:

```python
# Fancy indexing with arrays of indices
arr = np.arange(10)
print("Original Array:", arr)

# Fancy indexing to reorder elements
indices = [3, 1, 9, 7]
result = arr[indices]
print("Fancy Indexing Result:", result)

# Multi-indexing with slices
multi_indexed_result = arr[::2]  # Access every second element
print("Multi-Indexing Result:", multi_indexed_result)
```

```
Original Array: [0 1 2 3 4 5 6 7 8 9]
Fancy Indexing Result: [3 1 9 7]
Multi-Indexing Result: [0 2 4 6 8]
```

In [73]:

```python
# Stacking arrays vertically and horizontally
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

# Vertical stacking
vstacked = np.vstack((arr1, arr2))
print("Vertically Stacked Arrays:\n", vstacked)

# Horizontal stacking
hstacked = np.hstack((arr1, arr2))
print("Horizontally Stacked Arrays:\n", hstacked)

# Splitting arrays
split_arr = np.hsplit(hstacked, 2)
print("Horizontally Split Arrays:", split_arr)
```

```
Vertically Stacked Arrays:
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]
Horizontally Stacked Arrays:
 [[1 2 5 6]
 [3 4 7 8]]
Horizontally Split Arrays: [array([[1, 2],
       [3, 4]]), array([[5, 6],
       [7, 8]])]
```

In [74]:

```python
from numpy.linalg import inv, eig, solve

# Define a matrix
matrix = np.array([[1, 2], [3, 4]])

# Inverse of a matrix
matrix_inv = inv(matrix)
print("Inverse of Matrix:\n", matrix_inv)

# Eigenvalues and eigenvectors
eigenvalues, eigenvectors = eig(matrix)
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)

# Solving a system of linear equations
# For example, solving Ax = b, where A is the matrix, and b is a vector
A = np.array([[3, 1], [1, 2]])
b = np.array([9, 8])
x = solve(A, b)
print("Solution of Linear Equations:", x)
```

```
Inverse of Matrix:
 [[-2.   1. ]
 [ 1.5 -0.5]]
Eigenvalues: [-0.37228132  5.37228132]
Eigenvectors:
 [[-0.82456484 -0.41597356]
 [ 0.56576746 -0.90937671]]
Solution of Linear Equations: [2. 3.]
```

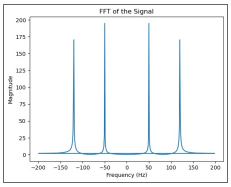
In [75]:

```python
# Binning an array of data
data = np.random.randn(1000)  # Generate 1000 random numbers
bins = np.linspace(-3, 3, 7)  # Create 6 bins between -3 and 3
digitized = np.digitize(data, bins)

# Counting the number of elements in each bin
bin_counts = np.bincount(digitized)
print("Bin Counts:", bin_counts)
```

```
Bin Counts: [  1  26 151 307 343 149  23]
```

In [77]:

```python
# Generate a sample signal
time = np.linspace(0, 1, 400)
signal = np.sin(2 * np.pi * 50 * time) + np.sin(2 * np.pi * 120 * time)

# Perform FFT
signal_fft = np.fft.fft(signal)
frequencies = np.fft.fftfreq(len(signal), d=time[1] - time[0])

# Plot the FFT result
import matplotlib.pyplot as plt
plt.plot(frequencies, np.abs(signal_fft))
plt.title('FFT of the Signal')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.show()
```



In [78]:

```python
from scipy import optimize

# Define a simple quadratic function
def f(x):
    return x**2 + 10*np.sin(x)

# Find the minimum of the function
result = optimize.minimize(f, x0=0)  # Start the search at x=0
print("Function minimum:", result.x)

# Numerical integration
from scipy.integrate import quad
```

```
result, error = quad(np.sin, 0, np.pi)
print("Integral of sin(x) from 0 to pi:", result)
```

```
Function minimum: [-1.30644012]
Integral of sin(x) from 0 to pi: 2.0
```

In [79]:

```
# Normalizing data
data = np.random.rand(100, 3)  # Random dataset with 3 features
normalized_data = (data - np.mean(data, axis=0)) / np.std(data, axis=0)
print("Normalized Data:\n", normalized_data)

# Generating polynomial features
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
poly_features = poly.fit_transform(data)
print("Polynomial Features:\n", poly_features)
```

```
Normalized Data:
 [[-0.07867606  1.54618142 -1.29728695]
 [-0.10420612 -1.27499644  0.53398214]
 [ 1.50242555 -0.28922969 -0.06329412]
 [-1.11531425 -1.24799388 -0.51755026]
 [ 1.35492871 -1.26585454  0.44063684]
 [ 1.35306439 -1.06624746  0.48733713]
 [-1.55077168 -1.39048182  1.92834303]
 [-0.05628639 -0.86397572  0.6636992 ]
 [-0.51667831  0.7977934   1.76684069]
 [ 0.99008777 -0.06148516 -0.26800809]
 [-0.46113287 -1.23992938 -0.25505175]
 [-0.95954609  0.11906083  0.67255312]
 [-1.59912787  0.06741177 -1.06307189]
 [-0.97347682  0.57885872  0.51008799]
 [ 0.25459418  0.16654153 -1.58046817]
 [ 0.5041768  -0.29322821  0.82043648]
 [-0.6093574   0.63011243 -0.66525166]
 [-0.18071358  0.40110395  1.64799404]
 [-0.43537875  0.33215211  0.42583036]
 [-0.92600611  0.21720443 -1.45598939]
 [-0.05346763 -0.42508215  1.49828225]
 [ 0.91533587  1.28506151  1.02704189]
 [-0.55417338 -0.53244166 -1.23664048]
 [-0.44137591  1.14908844 -0.07498698]
 [ 1.04614549  0.74272252  0.80511854]
 [-1.28817554  1.41955168 -0.47810675]
 [-1.00221074  1.40352547  0.89215949]
 [ 0.07416694 -1.5301512  -1.07390482]
 [-0.51508739  0.223562   -0.09120319]
 [-1.24062298  0.46283781  0.78932898]
 [-0.31027573  1.33046387 -0.79222998]
 [-0.56744718 -0.54464557  0.85576963]
 [ 0.82079411 -0.93264093  0.83734565]
 [ 0.09029362  0.06095115  1.35513143]
 [-1.11100974 -0.98067538 -0.76332469]
 [-1.06204529 -1.40980879  0.79301311]
 [ 0.90490182  1.16839234  1.41626827]
 [-1.67586776  0.58698027 -1.64810253]
 [ 1.27820901  1.17631363  1.86175951]
 [ 1.07748639 -1.27403474  0.36061581]
 [ 0.68940796  0.94352287 -0.54891891]
```

```
 [-0.9583129   0.94053006 -0.24575884]
 [ 1.28847543  1.65583783 -0.55437477]
 [-0.73414333  1.62686212  1.01402331]
 [ 1.44931944 -0.88037475 -0.93627462]
 [-0.61603356 -1.38947094 -1.29045207]
 [ 0.95026891 -0.69322642 -0.06243894]
 [ 1.00933614 -0.75955669 -1.15800272]
 [-0.61682381  0.46518041 -1.17582128]
 [-0.92588471  1.01377516  0.32366054]
 [-1.23839256  1.38691132  0.01745632]
 [ 1.50539223 -1.20861454  0.26728214]
 [ 1.4532346   1.65117695  1.61900853]
 [-1.44980081  0.44703516 -0.69439922]
 [-0.53740531  0.75766779  1.32930537]
 [ 0.27272659 -0.82622964  0.31405507]
 [-1.62728456  1.23041443 -0.16932265]
 [ 0.29366117  1.266253    1.65489094]
 [ 0.88696677 -0.81814617  1.08221023]
 [ 1.43176692  0.31444075 -1.55874555]
 [-1.11111929 -0.25382396  0.78451901]
 [-0.50978095  0.21930307  1.24785109]
 [ 0.76290696 -0.83431869 -0.03752822]
 [ 1.55685624  1.09443153 -1.44459393]
 [ 1.39221015 -1.37237802 -0.22068847]
 [ 1.27823625 -1.36676266  0.48330412]
 [-1.10990268 -0.017659    0.40519042]
 [-0.12278711 -1.15943009 -1.00601121]
 [-1.08304343 -0.73454301  0.37557261]
 [ 1.16121232  0.91959139 -1.62398821]
 [ 1.01761182 -1.3475367   1.46096009]
 [ 1.02449053 -0.88149665  0.6838838 ]
 [ 0.60425513 -0.98284355 -0.9986486 ]
 [ 0.77968183  1.00896766  0.23696891]
 [ 0.49948879 -1.27689378 -1.2558867 ]
 [-1.50526958 -0.12582776  0.41495445]
 [ 0.24423009 -1.38867114 -0.87197721]
 [ 1.25160994 -1.54205501 -0.05719881]
 [ 0.74645546  0.09373551  0.63144037]
 [ 1.4907586   0.23038493  0.33225524]
 [-0.38264888 -0.2689569  -0.40806976]
 [-1.49229964 -0.7215264  -1.57764736]
 [ 0.37020804  1.32955131 -1.19011621]
 [-0.39144937  0.95317271 -0.87230193]
 [-1.24783222  1.16285276 -0.61707654]
 [-1.0089783   0.14285115 -1.65867203]
 [-1.70312002  1.16327032 -1.45238822]
 [-0.92027631 -0.55061517 -0.10209376]
 [ 1.48219261  0.93961219  0.73998518]
 [-1.02163131 -1.39762379 -0.14760248]
 [-0.07641411 -1.45850756  0.71743082]
 [-0.63483676  0.14295218 -1.12512989]
 [ 0.75607758  0.75011468  1.61658544]
 [ 1.346191   -1.27263586 -0.13500214]
 [-0.04575234 -1.25811399 -1.45197508]
 [ 1.11362445 -0.16836523  1.81270239]
 [ 1.24647004  1.52636267 -1.14949633]
 [-1.59595063  1.58891332 -0.75772182]
 [ 0.74170811  1.23010978  0.88585877]
 [-0.20803671 -0.48254948 -0.96013454]]
Polynomial Features:
 [[1.00000000e+00 4.93365879e-01 9.64702515e-01 1.32791873e-01
  2.43409891e-01 4.75951305e-01 6.55149790e-02 9.30650942e-01
  1.28104653e-01 1.76336814e-02]
 [1.00000000e+00 4.85755180e-01 1.01855633e-01 6.21718442e-01
```

 2.35958095e-01 4.94769013e-02 3.02002954e-01 1.03745699e-02
 6.33255254e-02 3.86533821e-01]
[1.00000000e+00 9.64703865e-01 4.03348771e-01 4.62252946e-01
 9.30653548e-01 3.89112119e-01 4.45937204e-01 1.62690231e-01
 1.86449158e-01 2.13677786e-01]
[1.00000000e+00 1.84336429e-01 1.10114266e-01 3.40972084e-01
 3.39799191e-02 2.02980706e-02 6.28535764e-02 1.21251516e-02
 3.75458907e-02 1.16261962e-01]
[1.00000000e+00 9.20733978e-01 1.04651650e-01 5.96796382e-01
 8.47751058e-01 9.63563301e-02 5.49490706e-01 1.09519679e-02
 6.24557261e-02 3.56165921e-01]
[1.00000000e+00 9.20178211e-01 1.65700739e-01 6.09264791e-01
 8.46727939e-01 1.52474210e-01 5.60632186e-01 2.74567349e-02
 1.00955626e-01 3.71203586e-01]
[1.00000000e+00 5.45233757e-02 6.65348519e-02 9.93995836e-01
 2.97279850e-03 3.62770473e-03 5.41960084e-02 4.42688651e-03
 6.61353657e-02 9.88027721e-01]
[1.00000000e+00 5.00040404e-01 2.27564809e-01 6.56351319e-01
 2.50040406e-01 1.13791599e-01 3.28202179e-01 5.17857423e-02
 1.49362462e-01 4.30797054e-01]
[1.00000000e+00 3.62794196e-01 7.35810786e-01 9.50876677e-01
 1.31619629e-01 2.66947883e-01 3.44972540e-01 5.41417514e-01
 6.99665316e-01 9.04166455e-01]
[1.00000000e+00 8.11972218e-01 4.73003598e-01 4.07596806e-01
 6.59298882e-01 3.84065780e-01 3.30957283e-01 2.23732403e-01
 1.92794756e-01 1.66135157e-01]
[1.00000000e+00 3.79352697e-01 1.12580763e-01 4.11055991e-01
 1.43908469e-01 4.27078160e-02 1.55935199e-01 1.26744281e-02
 4.62769970e-02 1.68967028e-01]
[1.00000000e+00 2.30772061e-01 5.28222925e-01 6.58715207e-01
 5.32557441e-02 1.21899093e-01 1.52013066e-01 2.79019459e-01
 3.47948474e-01 4.33905724e-01]
[1.00000000e+00 4.01080421e-02 5.12426251e-01 1.95324445e-01
 1.60865504e-03 2.05524136e-02 7.83408108e-03 2.62580662e-01
 1.00089373e-01 3.81516390e-02]
[1.00000000e+00 2.26619208e-01 6.68850421e-01 6.15338994e-01
 5.13562653e-02 1.51574353e-01 1.39447635e-01 4.47360886e-01
 4.11569745e-01 3.78642077e-01]
[1.00000000e+00 5.92716182e-01 5.42744722e-01 5.71859322e-02
 3.51312472e-01 3.21693579e-01 3.38950274e-02 2.94571833e-01
 3.10373628e-02 3.27023084e-03]
[1.00000000e+00 6.67118592e-01 4.02125838e-01 6.98198266e-01
 4.45047215e-01 2.68265623e-01 4.65781044e-01 1.61705189e-01
 2.80763563e-01 4.87480819e-01]
[1.00000000e+00 3.35165880e-01 6.84526178e-01 3.01537606e-01
 1.12336167e-01 2.29429819e-01 1.01065117e-01 4.68576088e-01
 2.06410385e-01 9.09249281e-02]
[1.00000000e+00 4.62947747e-01 6.14484776e-01 9.19146067e-01
 2.14320616e-01 2.84474343e-01 4.25516601e-01 3.77591540e-01
 5.64801265e-01 8.44829493e-01]
[1.00000000e+00 3.87030190e-01 5.93396110e-01 5.92843232e-01
 1.49792368e-01 2.29662209e-01 2.29448229e-01 3.52118944e-01
 3.51790868e-01 3.51463098e-01]
[1.00000000e+00 2.40770575e-01 5.58239786e-01 9.04202528e-02
 5.79704700e-02 1.34407714e-01 2.17705363e-02 3.11631659e-01
 5.04761826e-02 8.17582212e-03]
[1.00000000e+00 5.00880696e-01 3.61798796e-01 8.79174838e-01
 2.50881471e-01 1.81218032e-01 4.40361705e-01 1.30898368e-01
 3.18084398e-01 7.72948397e-01]
[1.00000000e+00 7.89688126e-01 8.84839950e-01 7.53359396e-01
 6.23607336e-01 6.98747602e-01 5.94918970e-01 7.82941737e-01
 6.66602491e-01 5.67550380e-01]
[1.00000000e+00 3.51616638e-01 3.28963283e-01 1.48983742e-01
 1.23634260e-01 1.15668964e-01 5.23851625e-02 1.08216842e-01

```
 4.90101809e-02 2.21961553e-02]
[1.00000000e+00 3.85242394e-01 8.43253084e-01 4.59131096e-01
 1.48411702e-01 3.24856837e-01 1.76876763e-01 7.11075764e-01
 3.87163712e-01 2.10801363e-01]
[1.00000000e+00 8.28683435e-01 7.18967560e-01 6.94108561e-01
 6.86716235e-01 5.95796507e-01 5.75196266e-01 5.16914352e-01
 4.99041538e-01 4.81786694e-01]
[1.00000000e+00 1.32805212e-01 9.25973272e-01 3.51503020e-01
 1.76372243e-02 1.22974076e-01 4.66814330e-02 8.57426500e-01
 3.25482401e-01 1.23554373e-01]
[1.00000000e+00 2.18053415e-01 9.21071716e-01 7.17347437e-01
 4.75472918e-02 2.00842833e-01 1.56420058e-01 8.48373106e-01
 6.60728435e-01 5.14587345e-01]
[1.00000000e+00 5.38929499e-01 2.38174864e-02 1.92432185e-01
 2.90445005e-01 1.28359460e-02 1.03707381e-01 5.67272660e-04
 4.58325096e-03 3.70301459e-02]
[1.00000000e+00 3.63268460e-01 5.60184225e-01 4.54801565e-01
 1.31963974e-01 2.03497260e-01 1.65215064e-01 3.13806366e-01
 2.54772662e-01 2.06844464e-01]
[1.00000000e+00 1.46980978e-01 6.33365851e-01 6.89892941e-01
 2.16034078e-02 9.30927321e-02 1.01401139e-01 4.01152301e-01
 4.36954630e-01 4.75952270e-01]
[1.00000000e+00 4.24324320e-01 8.98726093e-01 2.67635938e-01
 1.80051128e-01 3.81351338e-01 1.13564437e-01 8.07708590e-01
 2.40531401e-01 7.16289955e-02]
[1.00000000e+00 3.47659622e-01 3.25230760e-01 7.07631786e-01
 1.20867213e-01 1.13069603e-01 2.46014999e-01 1.05775048e-01
 2.30143624e-01 5.00742744e-01]
[1.00000000e+00 7.61504535e-01 2.06563808e-01 7.02712808e-01
 5.79889157e-01 1.57299276e-01 5.35118990e-01 4.26686067e-02
 1.45155033e-01 4.93805290e-01]
[1.00000000e+00 5.43736980e-01 5.10450293e-01 8.40955312e-01
 2.95649903e-01 2.77550701e-01 4.57258501e-01 2.60559502e-01
 4.29265885e-01 7.07205836e-01]
[1.00000000e+00 1.85619637e-01 1.91872646e-01 2.75353301e-01
 3.44546498e-02 3.56153311e-02 5.11109798e-02 3.68151124e-02
 5.28327665e-02 7.58194402e-02]
[1.00000000e+00 2.00216298e-01 6.06237696e-02 6.90876558e-01
 4.00865658e-02 1.21378667e-02 1.38324747e-01 3.67524144e-03
 4.18835413e-02 4.77310419e-01]
[1.00000000e+00 7.86577660e-01 8.49157113e-01 8.57278104e-01
 6.18704415e-01 6.67928015e-01 6.74315805e-01 7.21067802e-01
 7.27963800e-01 7.34925748e-01]
[1.00000000e+00 1.72313160e-02 6.71334367e-01 3.91283796e-02
 2.96918251e-04 1.15679746e-02 6.74233475e-04 4.50689832e-01
 2.62682260e-02 1.53103009e-03]
[1.00000000e+00 8.97863272e-01 8.51579808e-01 9.76218847e-01
 8.06158454e-01 7.64602233e-01 8.76511048e-01 7.25188170e-01
 8.31328259e-01 9.53003238e-01]
[1.00000000e+00 8.38026387e-01 1.02149764e-01 5.75431741e-01
 7.02288225e-01 8.56041977e-02 4.82226983e-01 1.04345743e-02
 5.87802165e-02 3.31121688e-01]
[1.00000000e+00 7.22337360e-01 7.80381612e-01 3.32597036e-01
 5.21771262e-01 5.63698793e-01 2.40247265e-01 6.08995460e-01
 2.59552611e-01 1.10620788e-01]
[1.00000000e+00 2.31139686e-01 7.79466271e-01 4.13537087e-01
 5.34255543e-02 1.80165589e-01 9.55848322e-02 6.07567667e-01
 3.22338211e-01 1.71012922e-01]
[1.00000000e+00 9.00923765e-01 9.98240524e-01 3.31140387e-01
 8.11663631e-01 8.99338611e-01 2.98332244e-01 9.96484143e-01
 3.30557753e-01 1.09653956e-01]
[1.00000000e+00 2.97966279e-01 9.89378408e-01 7.49883593e-01
 8.87839032e-02 2.94801402e-01 2.23440023e-01 9.78869633e-01
 7.41918635e-01 5.62325403e-01]
```

```
[1.00000000e+00 9.48872545e-01 2.22549225e-01 2.29177774e-01
 9.00359107e-01 2.11170849e-01 2.17460498e-01 4.95281573e-02
 5.10033359e-02 5.25224521e-02]
[1.00000000e+00 3.33175667e-01 6.68440285e-02 1.34616702e-01
 1.11006025e-01 2.22708038e-02 4.48510095e-02 4.46812414e-03
 8.99832267e-03 1.81216565e-02]
[1.00000000e+00 8.00101923e-01 2.79787855e-01 4.62481270e-01
 6.40163087e-01 2.23858801e-01 3.70032153e-01 7.82812438e-02
 1.29396643e-01 2.13888925e-01]
[1.00000000e+00 8.17710295e-01 2.59500985e-01 1.69979066e-01
 6.68650127e-01 2.12196627e-01 1.38993633e-01 6.73407610e-02
 4.41097351e-02 2.88928830e-02]
[1.00000000e+00 3.32940087e-01 6.34082325e-01 1.65221728e-01
 1.10849102e-01 2.11111425e-01 5.50089365e-02 4.02060395e-01
 1.04764177e-01 2.72982194e-02]
[1.00000000e+00 2.40806764e-01 8.01868014e-01 5.65565133e-01
 5.79878976e-02 1.93095242e-01 1.36191910e-01 6.42992311e-01
 4.53508590e-01 3.19863920e-01]
[1.00000000e+00 1.47645882e-01 9.15990339e-01 4.83812330e-01
 2.17993064e-02 1.35242201e-01 7.14328982e-02 8.39038301e-01
 4.43167420e-01 2.34074371e-01]
[1.00000000e+00 9.65588256e-01 1.22158294e-01 5.50512785e-01
 9.32360680e-01 1.17954614e-01 5.31568680e-01 1.49226487e-02
 6.72497026e-02 3.03064327e-01]
[1.00000000e+00 9.50039684e-01 9.96815010e-01 9.11407288e-01
 9.02575402e-01 9.47013817e-01 8.65873092e-01 9.93640165e-01
 9.08504465e-01 8.30663244e-01]
[1.00000000e+00 8.46235321e-02 6.28532668e-01 2.93755563e-01
 7.16114219e-03 5.31886544e-02 2.48586333e-02 3.95053315e-01
 1.84634968e-01 8.62923309e-02]
[1.00000000e+00 3.56615326e-01 7.23538516e-01 8.34060068e-01
 1.27174490e-01 2.58024924e-01 2.97438603e-01 5.23507985e-01
 6.03474584e-01 6.95656197e-01]
[1.00000000e+00 5.98121586e-01 2.39109308e-01 5.63000589e-01
 3.57749431e-01 1.43016438e-01 3.36742805e-01 5.71732610e-02
 1.34618681e-01 3.16969664e-01]
[1.00000000e+00 3.17143237e-02 8.68126339e-01 4.33944619e-01
 1.00579833e-03 2.75320398e-02 1.37622601e-02 7.53643341e-01
 3.76718753e-01 1.88307932e-01]
[1.00000000e+00 6.04362339e-01 8.79087434e-01 9.20987455e-01
 3.65253837e-01 5.31287338e-01 5.56610132e-01 7.72794717e-01
 8.09628499e-01 8.48217892e-01]
[1.00000000e+00 7.81231089e-01 2.41581610e-01 7.68088671e-01
 6.10322014e-01 1.88731064e-01 6.00054749e-01 5.83616742e-02
 1.85556098e-01 5.89960207e-01]
[1.00000000e+00 9.43640011e-01 5.87979156e-01 6.29856063e-02
 8.90456470e-01 5.54840657e-01 5.94357382e-02 3.45719488e-01
 3.70342236e-02 3.96718659e-03]
[1.00000000e+00 1.85586980e-01 4.14177481e-01 6.88608737e-01
 3.44425271e-02 7.68659478e-02 1.27796816e-01 1.71542986e-01
 2.85206232e-01 4.74181992e-01]
[1.00000000e+00 3.64850349e-01 5.58881646e-01 8.12312767e-01
 1.33115777e-01 2.03908164e-01 2.96372596e-01 3.12348695e-01
 4.53986697e-01 6.59852031e-01]
[1.00000000e+00 7.44247951e-01 2.36635302e-01 4.69132127e-01
 5.53905013e-01 1.76115338e-01 3.49150625e-01 5.59962660e-02
 1.11013222e-01 2.20084953e-01]
[1.00000000e+00 9.80930054e-01 8.26536469e-01 9.34627016e-02
 9.62223772e-01 8.10774464e-01 9.16803729e-02 6.83162535e-01
 7.72503314e-02 8.73527659e-03]
[1.00000000e+00 9.31847847e-01 7.20718349e-02 4.20230570e-01
 8.68340409e-01 6.71599842e-02 3.91590952e-01 5.19434939e-03
 3.02867883e-02 1.76593732e-01]
[1.00000000e+00 8.97871392e-01 7.37892718e-02 6.08188027e-01
```

 8.06173036e-01 6.62532761e-02 5.46074630e-01 5.44485663e-03
 4.48777516e-02 3.69892677e-01]
[1.00000000e+00 1.85949658e-01 4.86407668e-01 5.87332618e-01
 3.45772753e-02 9.04473395e-02 1.09214300e-01 2.36592419e-01
 2.85683089e-01 3.44959605e-01]
[1.00000000e+00 4.80216051e-01 1.37201177e-01 2.10558953e-01
 2.30607456e-01 6.58862073e-02 1.01113789e-01 1.88241629e-02
 2.88889361e-02 4.43350727e-02]
[1.00000000e+00 1.93956598e-01 2.67151329e-01 5.79425025e-01
 3.76191620e-02 5.18157629e-02 1.12383307e-01 7.13698325e-02
 1.54794165e-01 3.35733360e-01]
[1.00000000e+00 8.62985699e-01 7.73062255e-01 4.55666112e-02
 7.44744317e-01 6.67141671e-01 3.93233339e-02 5.97625250e-01
 3.52258273e-02 2.07631606e-03]
[1.00000000e+00 8.20177338e-01 7.96694601e-02 8.69210276e-01
 6.72690866e-01 6.53430857e-02 7.12906570e-01 6.34722287e-03
 6.92495133e-02 7.55526503e-01]
[1.00000000e+00 8.22227931e-01 2.22206096e-01 6.61740361e-01
 6.76058770e-01 1.82704058e-01 5.44101407e-01 4.93755491e-02
 1.47042742e-01 4.37900305e-01]
[1.00000000e+00 6.96952676e-01 1.91209521e-01 2.12524679e-01
 4.85743033e-01 1.33263987e-01 1.48119644e-01 3.65610810e-02
 4.06367421e-02 4.51667392e-02]
[1.00000000e+00 7.49248663e-01 8.00397658e-01 5.42419523e-01
 5.61373560e-01 5.99696876e-01 4.06407103e-01 6.40636411e-01
 4.34151316e-01 2.94218939e-01]
[1.00000000e+00 6.65721061e-01 1.01275337e-01 1.43845237e-01
 4.43184531e-01 6.74211249e-02 9.57608038e-02 1.02566939e-02
 1.45679749e-02 2.06914522e-02]
[1.00000000e+00 6.80878857e-02 4.53324651e-01 5.89939495e-01
 4.63596018e-03 3.08659170e-02 4.01677329e-02 2.05503239e-01
 2.67434116e-01 3.48028608e-01]
[1.00000000e+00 5.89626569e-01 6.70886414e-02 2.46344398e-01
 3.47659491e-01 3.95572455e-02 1.45251202e-01 4.50088580e-03
 1.65269110e-02 6.06855624e-02]
[1.00000000e+00 8.89933893e-01 2.01767511e-02 4.63880320e-01
 7.91982333e-01 1.79559747e-02 4.12822819e-01 4.07101287e-04
 9.35959777e-03 2.15184951e-01]
[1.00000000e+00 7.39343636e-01 5.20477268e-01 6.47738604e-01
 5.46629012e-01 3.84811556e-01 4.78901415e-01 2.70896587e-01
 3.37133219e-01 4.19565299e-01]
[1.00000000e+00 9.61225862e-01 5.62270992e-01 5.67859814e-01
 9.23955158e-01 5.40469419e-01 5.45841539e-01 3.16148669e-01
 3.19291101e-01 3.22464768e-01]
[1.00000000e+00 4.02749352e-01 4.09549128e-01 3.70202045e-01
 1.62207041e-01 1.64945646e-01 1.49098634e-01 1.67730488e-01
 1.51615925e-01 1.37049554e-01]
[1.00000000e+00 7.19543192e-02 2.71132409e-01 5.79390543e-02
 5.17742405e-03 1.95091479e-02 4.16896520e-03 7.35127833e-02
 1.57091554e-02 3.35693401e-03]
[1.00000000e+00 6.27181523e-01 8.98446989e-01 1.61405156e-01
 3.93356662e-01 5.63489351e-01 1.01230331e-01 8.07206992e-01
 1.45013976e-01 2.60516242e-02]
[1.00000000e+00 4.00125861e-01 7.83332980e-01 2.46257704e-01
 1.60100704e-01 3.13431783e-01 9.85340758e-02 6.13610557e-01
 1.92901781e-01 6.06428568e-02]
[1.00000000e+00 1.44831851e-01 8.47462852e-01 3.14399776e-01
 2.09762651e-02 1.22739614e-01 4.55351016e-02 7.18193286e-01
 2.66442131e-01 9.88472192e-02]
[1.00000000e+00 2.16035958e-01 5.35499107e-01 3.63064516e-02
 4.66715350e-02 1.15687063e-01 7.84349904e-03 2.86759294e-01
 1.94420724e-02 1.31815843e-03]
[1.00000000e+00 9.10721842e-03 8.47590559e-01 9.13817199e-02
 8.29414274e-05 7.71919235e-03 8.32233283e-04 7.18409756e-01

```
 7.74542830e-02 8.35061873e-03]
[1.00000000e+00 2.42478670e-01 3.23404982e-01 4.51893915e-01
 5.87959054e-02 7.84188099e-02 1.09574635e-01 1.04590782e-01
 1.46144743e-01 2.04208110e-01]
[1.00000000e+00 9.58672279e-01 7.79185543e-01 6.76718746e-01
 9.19052539e-01 7.46983580e-01 6.48751503e-01 6.07130110e-01
 5.27289463e-01 4.57948261e-01]
[1.00000000e+00 2.12264003e-01 6.43505091e-02 4.39743641e-01
 4.50560070e-02 1.36592967e-02 9.33417455e-02 4.14098802e-03
 2.82977272e-02 1.93374470e-01]
[1.00000000e+00 4.94040183e-01 4.57294313e-02 6.70697008e-01
 2.44075702e-01 2.25921766e-02 3.31351273e-01 2.09118089e-03
 3.06705928e-02 4.49834477e-01]
[1.00000000e+00 3.27570294e-01 5.35530007e-01 1.78755714e-01
 1.07302298e-01 1.75423722e-01 5.85550618e-02 2.86792388e-01
 9.57290485e-02 3.19536052e-02]
[1.00000000e+00 7.42212061e-01 7.21228428e-01 9.10760352e-01
 5.50878744e-01 5.35304438e-01 6.75977318e-01 5.20170445e-01
 6.56866257e-01 8.29484419e-01]
[1.00000000e+00 9.18129202e-01 1.02577606e-01 4.43107778e-01
 8.42961231e-01 9.41794952e-02 4.06830191e-01 1.05221652e-02
 4.54529350e-02 1.96344503e-01]
[1.00000000e+00 5.03180681e-01 1.07019067e-01 9.14920232e-02
 2.53190798e-01 5.38499268e-02 4.60370186e-02 1.14530806e-02
 9.79139093e-03 8.37079032e-03]
[1.00000000e+00 8.48799407e-01 4.40314721e-01 9.63121193e-01
 7.20460433e-01 3.73738874e-01 8.17496697e-01 1.93877054e-01
 4.24076439e-01 9.27602432e-01]
[1.00000000e+00 8.88401653e-01 9.58641021e-01 1.72250171e-01
 7.89257497e-01 8.51658268e-01 1.53027336e-01 9.18992608e-01
 1.65126080e-01 2.96701213e-02]
[1.00000000e+00 4.10551994e-02 9.77771907e-01 2.76849199e-01
 1.68552940e-03 4.01426206e-02 1.13660991e-02 9.56037902e-01
 2.70695369e-01 7.66454791e-02]
[1.00000000e+00 7.37928417e-01 8.68033163e-01 7.15665220e-01
 5.44538348e-01 6.40546338e-01 5.28109703e-01 7.53481572e-01
 6.21221145e-01 5.12176708e-01]
[1.00000000e+00 4.54802519e-01 3.44222623e-01 2.22807464e-01
 2.06845331e-01 1.56553316e-01 1.01333396e-01 1.18489214e-01
 7.66953696e-02 4.96431660e-02]]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: