In [754…
```
# impor libraries for analysisdf_encoded
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Gathering

In [755…
```
url='/kaggle/input/remote-work-productivity/remote_work_productivity.csv'
df=pd.read_csv(url)
```

In [756…
```
df
```

Out[756…

	Employee_ID	Employment_Type	Hours_Worked_Per_Week	Productivity_Score	Well_Being_Score
0	1	Remote	29	75	78
1	2	In-Office	45	49	47
2	3	Remote	34	74	89
3	4	Remote	25	81	84
4	5	Remote	50	70	74
...	...	...	...	...	...
995	996	Remote	33	88	82
996	997	Remote	33	88	73
997	998	In-Office	45	74	61
998	999	In-Office	57	50	52
999	1000	Remote	34	80	82

1000 rows × 5 columns

Exploratory Data Analysis - EDA

In [757…
```
df.shape
```

Out[757… (1000, 5)

In [758…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Employee_ID            1000 non-null   int64
 1   Employment_Type        1000 non-null   object
 2   Hours_Worked_Per_Week  1000 non-null   int64
 3   Productivity_Score     1000 non-null   int64
 4   Well_Being_Score       1000 non-null   int64
dtypes: int64(4), object(1)
memory usage: 39.2+ KB
```

In [759…
```
df.index
```

Out[759… RangeIndex(start=0, stop=1000, step=1)

In [760…
```
df.dtypes
```

Out[760…
```
Employee_ID              int64
Employment_Type         object
Hours_Worked_Per_Week    int64
Productivity_Score       int64
Well_Being_Score         int64
dtype: object
```

In [761…
```
#convert categorical data to numeric
df_encoded=pd.get_dummies(df)
df_encoded
```

Out[761…

	Employee_ID	Hours_Worked_Per_Week	Productivity_Score	Well_Being_Score	Employment_Type_In-Office	Employment_Type_Remo
0	1	29	75	78	False	Tr
1	2	45	49	47	True	Fal
2	3	34	74	89	False	Tr
3	4	25	81	84	False	Tr
4	5	50	70	74	False	Tr
...	...	...	...	...	...	
995	996	33	88	82	False	Tr
996	997	33	88	73	False	Tr
997	998	45	74	61	True	Fal
998	999	57	50	52	True	Fal
999	1000	34	80	82	False	Tr

1000 rows × 6 columns

In [762…
```
df_encoded.columns
```

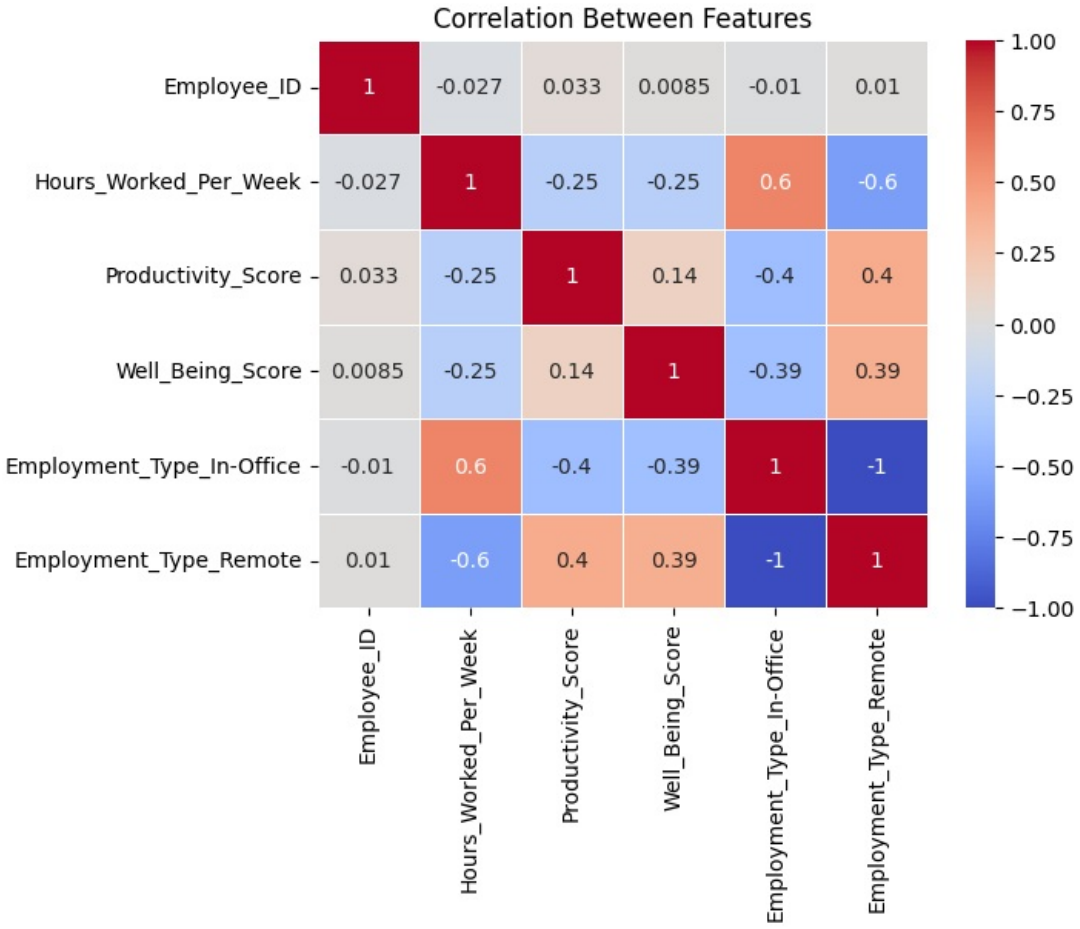
Out[762…
```
Index(['Employee_ID', 'Hours_Worked_Per_Week', 'Productivity_Score',
       'Well_Being_Score', 'Employment_Type_In-Office',
       'Employment_Type_Remote'],
      dtype='object')
```

In [763…
```
# calculate Correlation coefficient between features
corr_matrix = df_encoded.corr()
corr_matrix
```

Out[763…

	Employee_ID	Hours_Worked_Per_Week	Productivity_Score	Well_Being_Score	Employment_Type_In-Office	E
Employee_ID	1.000000	-0.027052	0.032728	0.008498	-0.010387	
Hours_Worked_Per_Week	-0.027052	1.000000	-0.254549	-0.251574	0.597426	
Productivity_Score	0.032728	-0.254549	1.000000	0.135163	-0.402885	
Well_Being_Score	0.008498	-0.251574	0.135163	1.000000	-0.390199	
Employment_Type_In-Office	-0.010387	0.597426	-0.402885	-0.390199	1.000000	
Employment_Type_Remote	0.010387	-0.597426	0.402885	0.390199	-1.000000	

In [764…
```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Between Features')
plt.show()
```

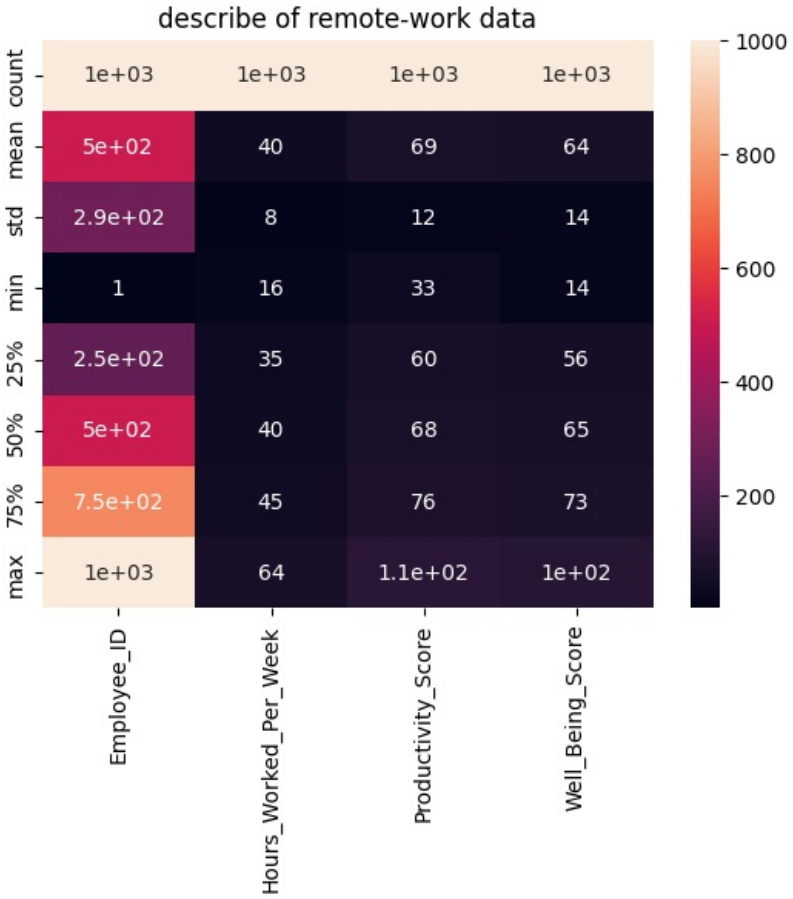


In [765…
```
# display the summary statistics of the dataset after encoding
df_encoded.describe()
```

Out[765…

	Employee_ID	Hours_Worked_Per_Week	Productivity_Score	Well_Being_Score
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	39.720000	68.602000	63.975000
std	288.819436	8.042779	12.235494	13.870572
min	1.000000	16.000000	33.000000	14.000000
25%	250.750000	34.750000	60.000000	56.000000
50%	500.500000	40.000000	68.000000	65.000000
75%	750.250000	45.000000	76.000000	73.000000
max	1000.000000	64.000000	112.000000	104.000000

In [766…
```
sns.heatmap(df_encoded.describe(),annot=True)
plt .title('describe of remote-work data')
plt.show()
```



. Data Cleaning & Visualization

In [767…
```
df_encoded.isna().sum()
```

Out[767…
```
Employee_ID                  0
Hours_Worked_Per_Week        0
Productivity_Score           0
Well_Being_Score             0
Employment_Type_In-Office    0
Employment_Type_Remote       0
dtype: int64
```

In [768…
```
df_encoded.duplicated().sum()
```

Out[768…
```
0
```

In [769…
```
df_encoded.columns
```

Out[769…
```
Index(['Employee_ID', 'Hours_Worked_Per_Week', 'Productivity_Score',
       'Well_Being_Score', 'Employment_Type_In-Office',
       'Employment_Type_Remote'],
      dtype='object')
```

In [770…
```
# handling outliers
# Step 1: Calculate Q1 and Q3
Q1=df_encoded['Hours_Worked_Per_Week'].quantile(0.25)
Q3=df_encoded['Hours_Worked_Per_Week'].quantile(0.75)

# Step 2: Calculate IQR
IQR = Q3 - Q1

# Step 3: Determine the lower and upper bounds
lower_bound =  Q1-1.5*IQR
upper_bound =  Q1+1.5*IQR

# Step 4: Remove outliers
df_encoded = df_encoded[(df_encoded['Hours_Worked_Per_Week'] >= lower_bound) & (df_encoded['Hours_Worked_Per_We
```

In [771…
```
Q1=df_encoded['Productivity_Score'].quantile(0.25)
Q3=df_encoded['Productivity_Score'].quantile(0.75)

IQR = Q3 - Q1


lower_bound =  Q1-1.5*IQR
```

```
upper_bound =  Q1+1.5*IQR

df_encoded = df_encoded[(df_encoded['Productivity_Score'] >= lower_bound) & (df_encoded['Productivity_Score'] <=
```

In [772…
```
from sklearn.preprocessing import MinMaxScaler
```

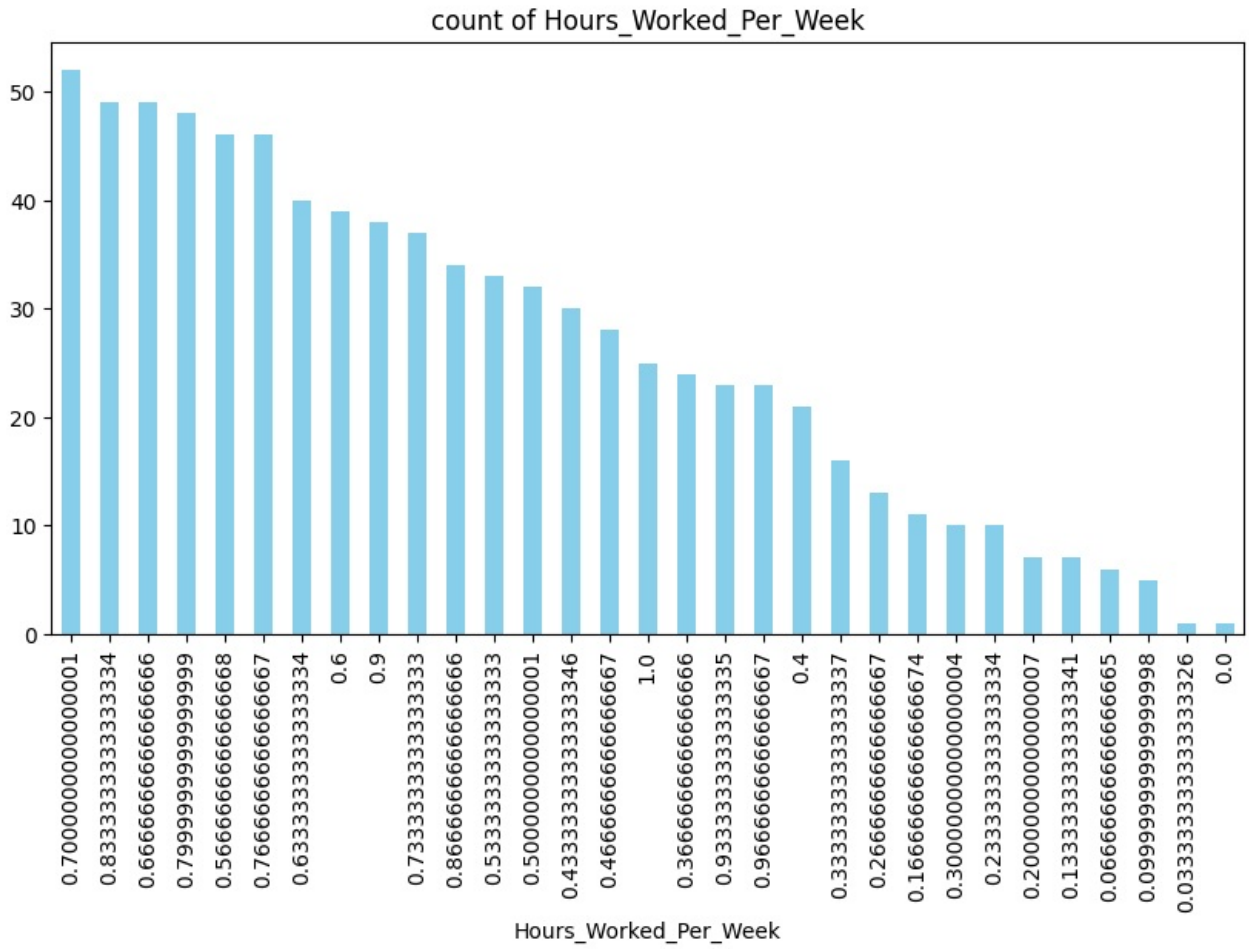
In [773…
```
scaler = MinMaxScaler()

df_encoded_scaled = scaler.fit_transform(df_encoded)
```

In [774…
```
df_encoded = pd.DataFrame(df_encoded_scaled, columns=df_encoded.columns)
```

In [775…
```
df_encoded['Hours_Worked_Per_Week'].value_counts()
```

Out[775…
```
Hours_Worked_Per_Week
0.700000    52
0.833333    49
0.666667    49
0.800000    48
0.566667    46
0.766667    46
0.633333    40
0.600000    39
0.900000    38
0.733333    37
0.866667    34
0.533333    33
0.500000    32
0.433333    30
0.466667    28
1.000000    25
0.366667    24
0.933333    23
0.966667    23
0.400000    21
0.333333    16
0.266667    13
0.166667    11
0.300000    10
0.233333    10
0.200000     7
0.133333     7
0.066667     6
0.100000     5
0.033333     1
0.000000     1
Name: count, dtype: int64
```

In [776…
```
df_encoded['Hours_Worked_Per_Week'].value_counts().plot(kind='bar' , figsize=(10,5),color='skyblue')
plt.title('count of Hours_Worked_Per_Week')
plt.show()
```

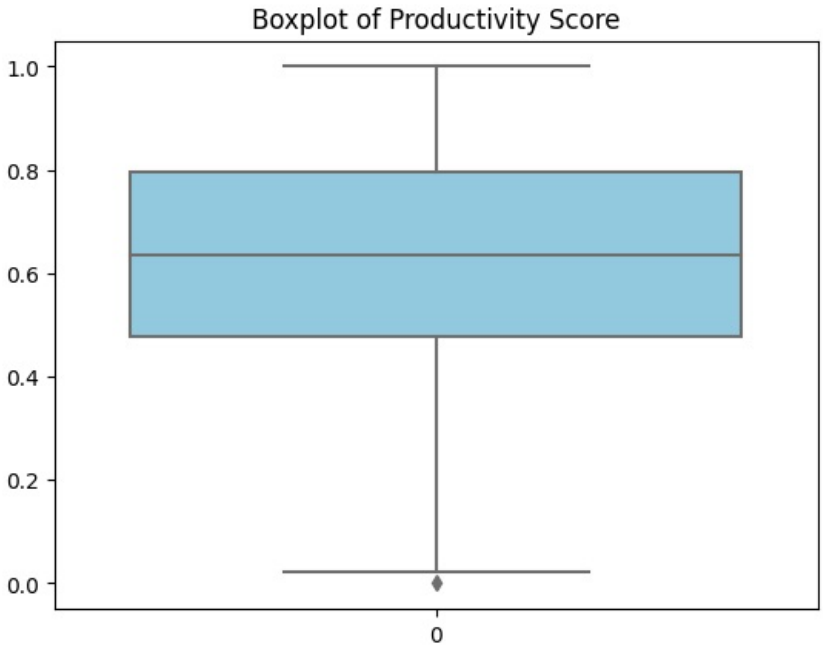


In [777…

```
df_encoded['Productivity_Score'].value_counts()
```

Out[777…
```
Productivity_Score
0.613636    38
0.795455    34
0.545455    33
0.772727    33
0.750000    33
0.590909    33
0.704545    32
0.636364    31
0.727273    30
0.818182    29
0.477273    29
0.681818    26
0.522727    26
0.659091    26
0.568182    25
0.863636    24
0.386364    23
0.840909    21
0.431818    20
0.909091    20
0.318182    19
0.886364    19
0.500000    18
1.000000    17
0.409091    16
0.363636    15
0.454545    14
0.954545    14
0.931818    14
0.227273    13
0.977273    12
0.272727    10
0.250000     8
0.295455     8
0.340909     8
0.204545     8
0.113636     7
0.181818     5
0.136364     3
0.022727     3
0.068182     2
0.090909     2
0.159091     2
0.000000     1
Name: count, dtype: int64
```

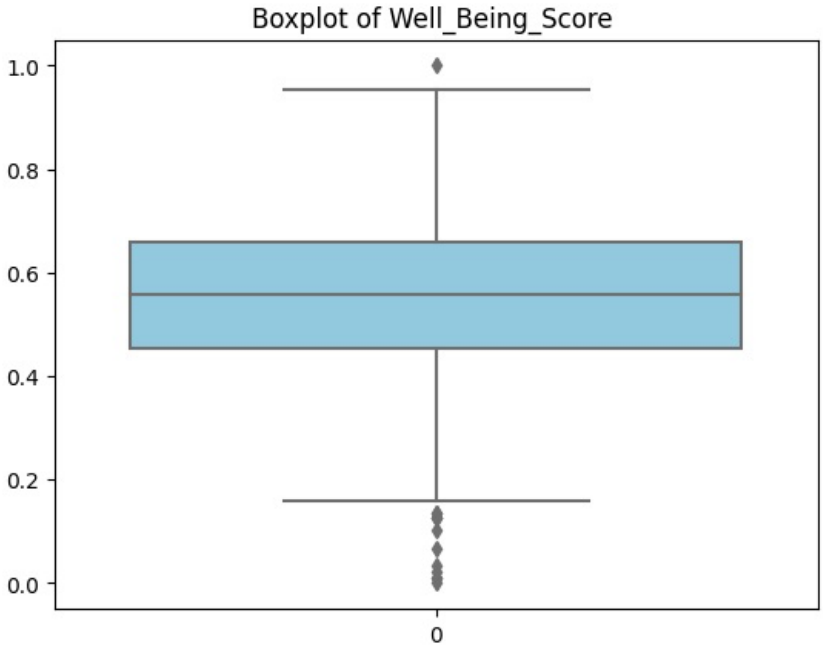
In [778…
```
sns.boxplot(df_encoded['Productivity_Score'],color='skyblue')
plt.title('Boxplot of Productivity Score')
plt.show()
```



In [779…
```
df_encoded['Well_Being_Score'].value_counts()
```

Out[779…
```
Well_Being_Score
0.568182    37
0.545455    29
0.602273    29
0.579545    28
0.511364    28
            ..
0.875000     1
0.102273     1
0.000000     1
0.034091     1
0.954545     1
Name: count, Length: 76, dtype: int64
```

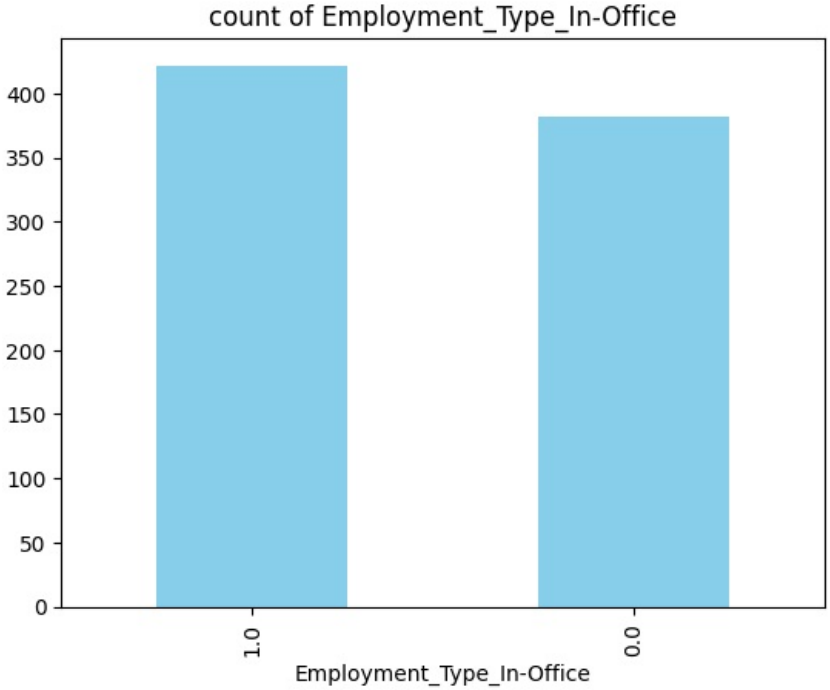
In [780…
```
sns.boxplot(df_encoded['Well_Being_Score'],color='skyblue')
plt.title('Boxplot of Well_Being_Score ')
plt.show()
```



In [781…
```
df_encoded['Employment_Type_In-Office'].value_counts()
```

Out[781…
```
Employment_Type_In-Office
1.0    422
0.0    382
Name: count, dtype: int64
```

In [782…
```
df_encoded['Employment_Type_In-Office'].value_counts().plot(kind='bar' , color='skyblue')
plt.title('count of Employment_Type_In-Office')
plt.show()
```



In [783…
```
# relation between Hours_Worked_Per_Week  Productivity_Score
hours_vs_productivity = df_encoded.groupby('Hours_Worked_Per_Week')['Productivity_Score'].mean().reset_index()
hours_vs_productivity
```

Out[783…

	Hours_Worked_Per_Week	Productivity_Score
0	0.000000	1.000000
1	0.033333	0.840909
2	0.066667	0.632576
3	0.100000	0.704545
4	0.133333	0.672078
5	0.166667	0.683884
6	0.200000	0.714286
7	0.233333	0.670455
8	0.266667	0.655594
9	0.300000	0.656818
10	0.333333	0.619318
11	0.366667	0.666667
12	0.400000	0.643939
13	0.433333	0.672727
14	0.466667	0.640422
15	0.500000	0.617898
16	0.533333	0.677686
17	0.566667	0.624012
18	0.600000	0.632867
19	0.633333	0.631818
20	0.666667	0.585343
21	0.700000	0.578234
22	0.733333	0.628993
23	0.766667	0.631917
24	0.800000	0.589015
25	0.833333	0.546382
26	0.866667	0.589572
27	0.900000	0.656699
28	0.933333	0.555336
29	0.966667	0.580040
30	1.000000	0.614545

In [784…
```
# relation between Well_Being_Score  Productivity_Score
Well_Being_Score_vs_productivity = df_encoded.groupby('Well_Being_Score')['Productivity_Score'].mean().reset_in
Well_Being_Score_vs_productivity
```

Out[784…

	Well_Being_Score	Productivity_Score
0	0.000000	0.454545
1	0.011364	0.397727
2	0.022727	0.431818
3	0.034091	0.704545
4	0.068182	0.204545
...	...	...
71	0.886364	0.625000
72	0.897727	0.738636
73	0.920455	0.659091
74	0.954545	0.659091
75	1.000000	0.363636

76 rows × 2 columns

In [785…
```
# relation between Employment_Type_In-Office  Productivity_Score
Employment_Type_In_Office_vs_productivity = df_encoded.groupby('Employment_Type_In-Office')['Productivity_Score
Employment_Type_In_Office_vs_productivity
```

Out[785…

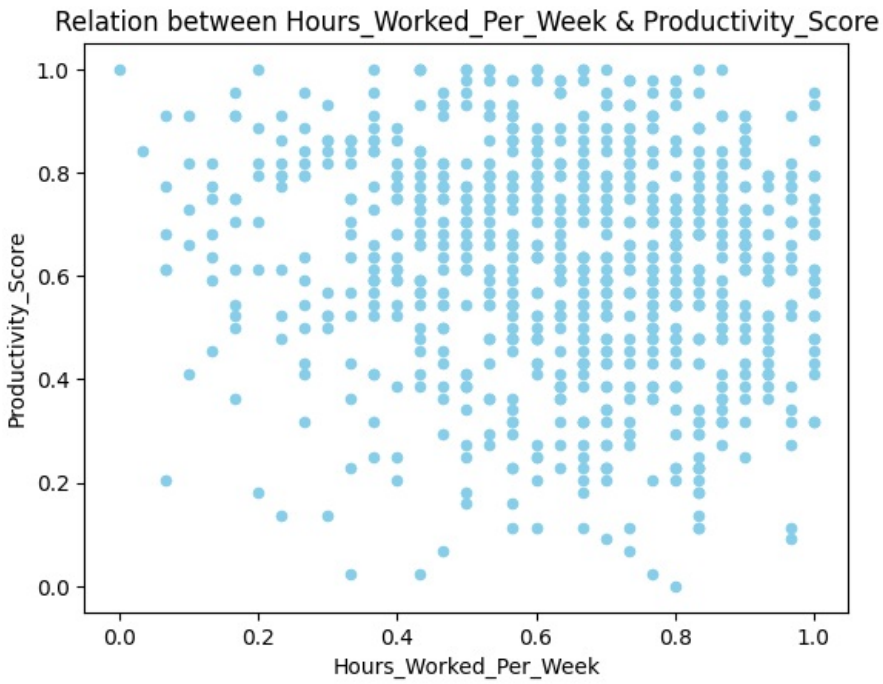
	Employment_Type_In-Office	Productivity_Score
0	0.0	0.687411
1	1.0	0.560534

In [786…
```
# relation between Employment_Type_Remote  Productivity_Score
Employment_Type_Remote_vs_productivity = df_encoded.groupby('Employment_Type_Remote')['Productivity_Score'].mea
Employment_Type_Remote_vs_productivity
```

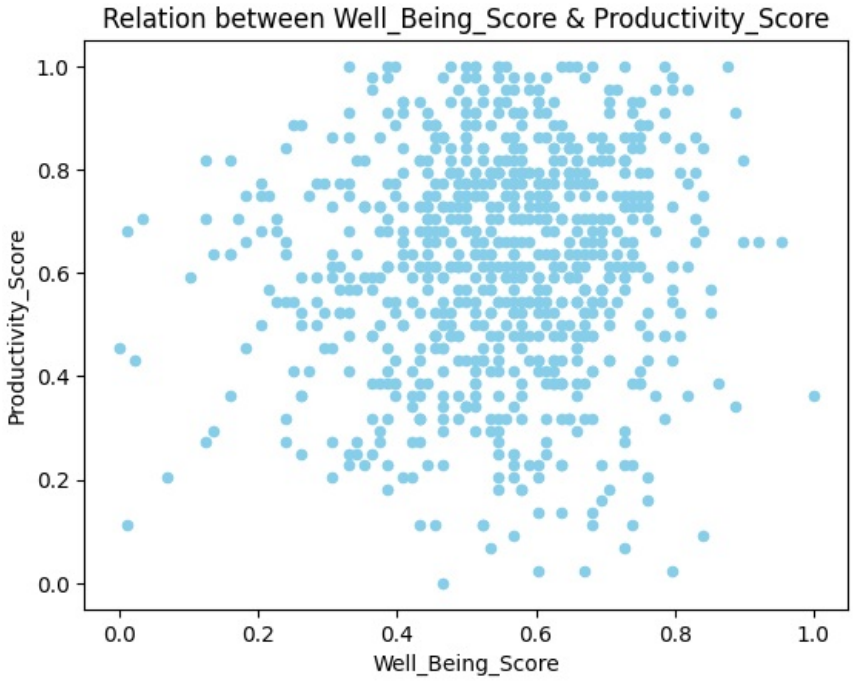
Out[786…

	Employment_Type_Remote	Productivity_Score
0	0.0	0.560534
1	1.0	0.687411

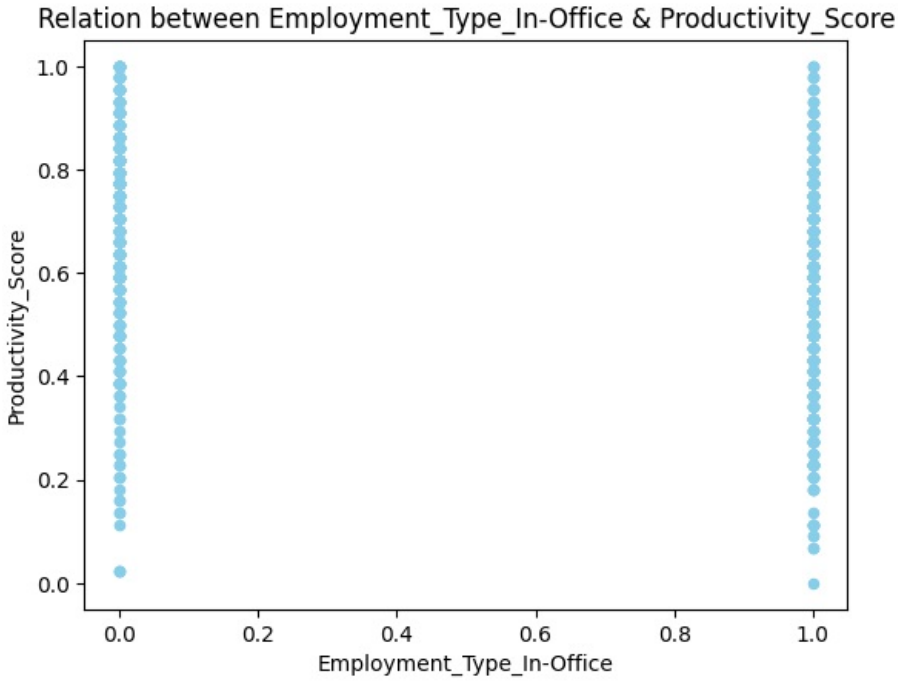
In [787…
```
df_encoded.plot(kind='scatter', x='Hours_Worked_Per_Week', y='Productivity_Score',color='skyblue')
plt.title('Relation between Hours_Worked_Per_Week & Productivity_Score')
plt.show()
```



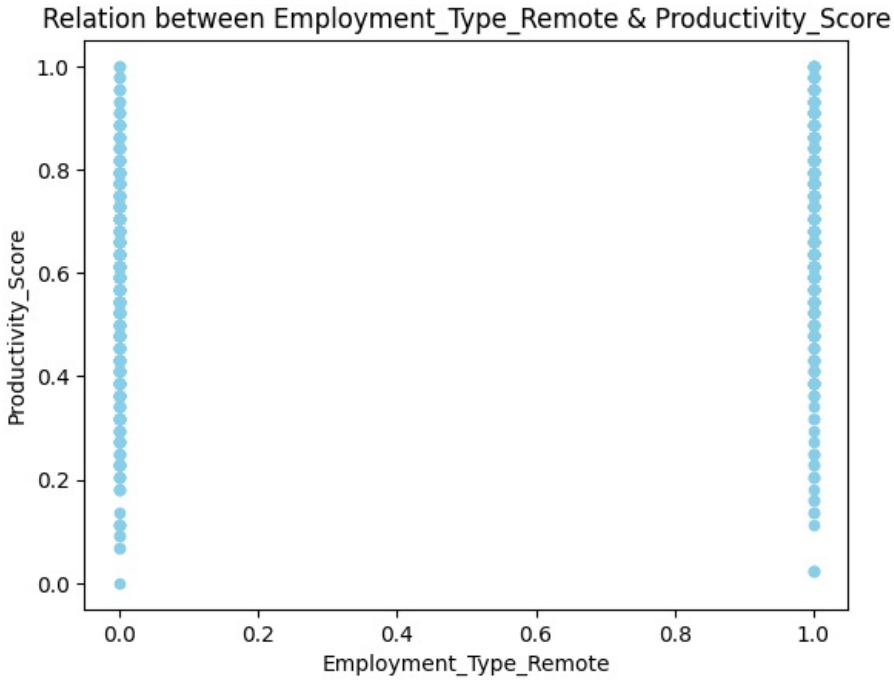
In [788…
```
df_encoded.plot(kind='scatter', x='Well_Being_Score', y='Productivity_Score',color='skyblue')
plt.title('Relation between Well_Being_Score & Productivity_Score')
plt.show()
```



In [789…
```
df_encoded.plot(kind='scatter', x='Employment_Type_In-Office', y='Productivity_Score',color='skyblue')
plt.title('Relation between Employment_Type_In-Office & Productivity_Score')
plt.show()
```



In [790…
```
df_encoded.plot(kind='scatter', x='Employment_Type_Remote', y='Productivity_Score',color='skyblue')
plt.title('Relation between Employment_Type_Remote & Productivity_Score')
plt.show()
```



Modeling

In [791…
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [792…
```
df_encoded.columns
```

Out[792…
```
Index(['Employee_ID', 'Hours_Worked_Per_Week', 'Productivity_Score',
       'Well_Being_Score', 'Employment_Type_In-Office',
       'Employment_Type_Remote'],
      dtype='object')
```

In [793…
```
X = df_encoded.drop(columns=['Employee_ID' , 'Productivity_Score'])
Y = df_encoded['Productivity_Score']
```

In [794…
```
X_train,X_test,Y_train,Y_test=train_test_split(X , Y , test_size=0.2 , random_state=42)
```

In [795…
```
model = LinearRegression()
model.fit(X_train , Y_train)
```

Out[795…
```
▾ LinearRegression
LinearRegression()
```

In [796…
```
y_pre = model.predict(X_test)
```

In [797…
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [798…
```
MSE = mean_squared_error(Y_test , y_pre)
MSE
```

Out[798…
```
0.045883745073734626
```

In [799…
```
R2 = r2_score(Y_test , y_pre)
R2
```

Out[799…
```
0.05053537132054586
```

In [800…
```
model.predict(X)
```

Out[800…
```
array([0.67254009, 0.56694872, 0.67119828, 0.6630888 , 0.70260328,
       0.56301465, 0.68562749, 0.67178328, 0.66601377, 0.55335677,
       0.70173124, 0.68153253, 0.70139852, 0.68583408, 0.55602948,
       0.69838217, 0.54449047, 0.55628175, 0.70879687, 0.56067467,
       0.69136223, 0.58769766, 0.53992573, 0.56397806, 0.56054854,
       0.55640788, 0.56285375, 0.68596022, 0.71687159, 0.56519373,
       0.54611932, 0.55348291, 0.68768044, 0.6875543 , 0.69637492,
       0.69596174, 0.6850425 , 0.55602948, 0.52119496, 0.5546529 ,
       0.71197412, 0.69190154, 0.54147413, 0.68453796, 0.54406638,
       0.67990368, 0.51843089, 0.53575031, 0.69525061, 0.70294692,
```

0.68453796, 0.69784286, 0.69357607, 0.66329539, 0.56552645,
0.53394964, 0.54637159, 0.5629342 , 0.69917376, 0.53984528,
0.67304464, 0.55414835, 0.57355549, 0.54586705, 0.68726726,
0.56837098, 0.55963082, 0.68056912, 0.55544448, 0.68826543,
0.68412478, 0.54908998, 0.55602948, 0.54766772, 0.55628175,
0.54820703, 0.54164595, 0.54574091, 0.55565107, 0.55569675,
0.55783015, 0.54720886, 0.7027751 , 0.54787431, 0.55226723,
0.55351767, 0.56209693, 0.54319435, 0.55185406, 0.55021429,
0.66865171, 0.54980111, 0.69604219, 0.57134164, 0.54837885,
0.68872429, 0.52462448, 0.66961512, 0.6536375 , 0.68516864,
0.54787431, 0.56966711, 0.70193783, 0.54624546, 0.56620282,
0.6875543 , 0.67279236, 0.68722158, 0.68425092, 0.68742817,
0.69512447, 0.68228934, 0.68374637, 0.55130383, 0.68964201,
0.56343875, 0.54486888, 0.55808242, 0.6865909 , 0.67630234,
0.55030566, 0.70348623, 0.68780657, 0.68240456, 0.68768044,
0.5723855 , 0.69207336, 0.56372578, 0.68153253, 0.5629342 ,
0.69307153, 0.5571647 , 0.54189822, 0.69370221, 0.57987522,
0.6777246 , 0.69687946, 0.53148352, 0.69002042, 0.68888519,
0.57075665, 0.57723728, 0.56824484, 0.67245964, 0.70440395,
0.67329691, 0.70231624, 0.67663506, 0.56418464, 0.67998413,
0.71344207, 0.54812658, 0.69784286, 0.67496052, 0.5687037 ,
0.67688733, 0.67450167, 0.70210965, 0.54791999, 0.55531834,
0.69057065, 0.67638279, 0.55691243, 0.68885043, 0.68186525,
0.57519525, 0.54611932, 0.56837098, 0.69156882, 0.68508818,
0.68425092, 0.69366744, 0.54624546, 0.68839157, 0.57677842,
0.68788702, 0.57167436, 0.55473335, 0.5563622 , 0.55615561,
0.67622189, 0.54833317, 0.70256851, 0.68270252, 0.55808242,
0.5629342 , 0.55440063, 0.55180837, 0.6625038 , 0.56205125,
0.55795628, 0.69562902, 0.68579932, 0.55427449, 0.69499834,
0.69018132, 0.67630234, 0.69921944, 0.54716318, 0.6746278 ,
0.57665228, 0.68153253, 0.70076784, 0.69022701, 0.5531045 ,
0.56125967, 0.66417834, 0.67571734, 0.55335677, 0.68224366,
0.68776089, 0.70013716, 0.55891969, 0.52892603, 0.55193451,
0.68006458, 0.57368162, 0.54921612, 0.58309815, 0.68036254,
0.57243118, 0.67320554, 0.57372731, 0.54348138, 0.56958666,
0.68951587, 0.68529477, 0.70615893, 0.69365653, 0.53796415,
0.69319767, 0.5816302 , 0.55005339, 0.56460874, 0.54725455,
0.67479962, 0.69632923, 0.56205125, 0.67015443, 0.69474607,
0.66321493, 0.54403161, 0.55490517, 0.66806672, 0.55444631,
0.56786644, 0.55272609, 0.54962929, 0.54720886, 0.68353979,
0.68867861, 0.69599651, 0.54720886, 0.70198351, 0.69273881,
0.68027117, 0.56707485, 0.6808214 , 0.55084497, 0.56644417,
0.55874786, 0.53913414, 0.68458364, 0.55553585, 0.55130383,
0.67751801, 0.55833469, 0.69278449, 0.55272609, 0.56857757,
0.69959785, 0.5702521 , 0.55176269, 0.68115412, 0.56489577,
0.67421463, 0.58292633, 0.67287282, 0.67479962, 0.5556163 ,
0.54131322, 0.67655461, 0.68847202, 0.70411691, 0.55051225,
0.56598531, 0.54093482, 0.70490849, 0.56347351, 0.68006458,
0.69198199, 0.55376995, 0.6850425 , 0.5521411 , 0.5525195 ,
0.69286494, 0.57469071, 0.56226875, 0.54758727, 0.55816287,
0.69972398, 0.56008968, 0.69499834, 0.54883771, 0.54675   ,
0.54256367, 0.55376995, 0.68236979, 0.69002042, 0.68684317,
0.67977754, 0.67990368, 0.67910118, 0.54210481, 0.566031 ,
0.70323396, 0.56151194, 0.56402374, 0.55971127, 0.68466409,
0.54720886, 0.69060541, 0.67876845, 0.56314079, 0.55067315,
0.55289791, 0.56954097, 0.54277025, 0.55607516, 0.69708605,
0.5386296 , 0.54637159, 0.70206397, 0.70645689, 0.67513235,
0.67596961, 0.57791364, 0.56573304, 0.56381715, 0.54871158,
0.5505927 , 0.71255911, 0.54335525, 0.68696931, 0.69746445,
0.68834588, 0.55929809, 0.56540032, 0.70678961, 0.57225936,
0.54746113, 0.55402222, 0.55013384, 0.67266623, 0.69261267,
0.69127086, 0.55071883, 0.68859816, 0.55222155, 0.69273881,
0.55381563, 0.68216321, 0.55034042, 0.56615713, 0.5579106 ,
0.54063686, 0.56720099, 0.66911057, 0.68696931, 0.56887552,
0.67007398, 0.68717589, 0.53666803, 0.69959785, 0.58648199,
0.68395296, 0.69570947, 0.68094753, 0.68483591, 0.57251163,
0.70482804, 0.56359965, 0.67517803, 0.53834256, 0.69269312,
0.5513952 , 0.67935345, 0.55348291, 0.68981383, 0.55427449,
0.5484593 , 0.56143149, 0.70541304, 0.56623759, 0.70407122,
0.5598374 , 0.7120198 , 0.54754158, 0.56347351, 0.69110996,
0.6802364 , 0.54415775, 0.66596809, 0.69624878, 0.68948111,
0.55034042, 0.56598531, 0.70043512, 0.68864384, 0.56849712,
0.69403493, 0.67848141, 0.56728144, 0.68074094, 0.54398593,
0.55557062, 0.68583408, 0.55038611, 0.5359569 , 0.55381563,
0.56088126, 0.54055641, 0.55549016, 0.68136071, 0.56004399,
0.66593332, 0.55996354, 0.6735035 , 0.68663658, 0.55578812,
0.69553765, 0.53031353, 0.6850425 , 0.7002633 , 0.55719947,
0.54540819, 0.69491789, 0.56933438, 0.70076784, 0.54858544,
0.6860059 , 0.53118556, 0.68801316, 0.56829053, 0.71327025,
0.55381563, 0.54022368, 0.67353826, 0.6940697 , 0.57878568,
0.6777246 , 0.69010087, 0.56385192, 0.54022368, 0.56268193,
0.55937854, 0.5488834 , 0.55841514, 0.69073155, 0.68976815,
0.67889459, 0.57096323, 0.56979324, 0.56941483, 0.6791121 ,

```
0.70310782, 0.57271822, 0.6844575 , 0.55900014, 0.67919255,
0.67333167, 0.54398593, 0.5488834 , 0.56423033, 0.55368949,
0.68031685, 0.57736341, 0.5505927 , 0.55762356, 0.56560691,
0.68128026, 0.68153253, 0.57138733, 0.67973186, 0.56631804,
0.54837885, 0.55800197, 0.66915626, 0.56184466, 0.5816302 ,
0.68922884, 0.58150407, 0.55808242, 0.67584348, 0.5525195 ,
0.5598374 , 0.55531834, 0.69286494, 0.547335  , 0.52382198,
0.54436434, 0.54239184, 0.69202768, 0.67973186, 0.54398593,
0.56669645, 0.68722158, 0.5571647 , 0.54695659, 0.6892636 ,
0.54540819, 0.68621249, 0.67743756, 0.55130383, 0.68370069,
0.67935345, 0.53616348, 0.5702521 , 0.69767104, 0.69106427,
0.54657818, 0.55021429, 0.56494146, 0.53712689, 0.55833469,
0.68554704, 0.55297836, 0.69156882, 0.70010239, 0.55038611,
0.69993057, 0.68366592, 0.55862173, 0.67329691, 0.71661932,
0.53575031, 0.68077571, 0.67906641, 0.55640788, 0.56628327,
0.70879687, 0.67279236, 0.6954572 , 0.66814717, 0.54996202,
0.56821008, 0.67881414, 0.68161298, 0.70081352, 0.56326692,
0.5708371 , 0.54561478, 0.56080081, 0.67513235, 0.55549016,
0.55495085, 0.5484593 , 0.5500077 , 0.57271822, 0.67413418,
0.68173911, 0.54043027, 0.69357607, 0.55891969, 0.57782227,
0.71222639, 0.66873216, 0.5344085 , 0.5638976 , 0.54871158,
0.70930142, 0.55427449, 0.53946687, 0.56598531, 0.54502978,
0.68102798, 0.56552645, 0.54314866, 0.55444631, 0.69985012,
0.56519373, 0.53909938, 0.55130383, 0.57535616, 0.69913899,
0.55185406, 0.69888672, 0.5525195 , 0.54160026, 0.54908998,
0.54385979, 0.57360117, 0.69482652, 0.55582289, 0.6646372 ,
0.67973186, 0.67889459, 0.69830172, 0.6875543 , 0.57514957,
0.68948111, 0.69244085, 0.58109089, 0.68128026, 0.68056912,
0.68554704, 0.56205125, 0.66865171, 0.5392146 , 0.69144268,
0.5384687 , 0.69942603, 0.69164927, 0.55188882, 0.52784741,
0.5270906 , 0.54632591, 0.53926028, 0.68470978, 0.68241548,
0.58137793, 0.54348138, 0.69570947, 0.68211752, 0.67228782,
0.55594902, 0.55485948, 0.68353979, 0.6933238 , 0.68885043,
0.53800984, 0.55051225, 0.68575363, 0.57573456, 0.56234921,
0.68876998, 0.68470978, 0.540178  , 0.54896385, 0.68759999,
0.67200078, 0.56966711, 0.54218526, 0.68086708, 0.56728144,
0.55920672, 0.67622189, 0.55653402, 0.67450167, 0.68981383,
0.68575363, 0.67785073, 0.67149624, 0.70059602, 0.584727  ,
0.69073155, 0.55874786, 0.53591121, 0.67396236, 0.68747385,
0.68199139, 0.54344662, 0.56054854, 0.67902073, 0.70152466,
0.70323396, 0.57665228, 0.67083079, 0.55820855, 0.57652615,
0.67342304, 0.54231139, 0.68579932, 0.55549016, 0.67739188,
0.56059422, 0.67685257, 0.67935345, 0.70185738, 0.57924454,
0.68796748, 0.55854128, 0.69792331, 0.67203555, 0.55134951,
0.69850831, 0.68265683, 0.68730203, 0.5363353 , 0.55343722,
0.67622189, 0.53529145, 0.68784134, 0.68316138, 0.68466409,
0.66974125, 0.68031685, 0.67696778, 0.56062899, 0.55862173,
0.54611932, 0.56966711, 0.70595235, 0.5619708 , 0.6970056 ,
0.56017013, 0.69353039, 0.55444631, 0.55277178, 0.70386464,
0.68278297, 0.53754006, 0.5490443 , 0.67023488, 0.68270252,
0.70930142, 0.55749742, 0.70056125, 0.57619342, 0.55293268,
0.6860059 , 0.68181957, 0.68621249, 0.67630234, 0.55849559,
0.54047596, 0.53570462, 0.69043359, 0.68885043, 0.68069526,
0.55569675, 0.55569675, 0.69248654, 0.69156882, 0.55544448,
0.52503765, 0.68262207, 0.69223426, 0.56130535, 0.68893088,
0.68140639, 0.66756217, 0.53557849, 0.54837885, 0.5525195 ,
0.53984528, 0.54256367, 0.55452676, 0.56573304, 0.69667287,
0.67325122, 0.6970056 , 0.68529477, 0.70516076, 0.69244085,
0.5521411 , 0.55419404, 0.67517803, 0.67274668, 0.5677403 ,
0.6704067 , 0.54502978, 0.66923671, 0.55131475, 0.68366592,
0.52834104, 0.55452676, 0.56377147, 0.67898596, 0.67190941,
0.70056125, 0.56740758, 0.56113353, 0.68521432, 0.57096323,
0.55389608, 0.539593  , 0.67588916, 0.68788702, 0.68491636,
0.5613858 , 0.67362963, 0.68734771, 0.56644417, 0.55013384,
0.68562749, 0.55419404, 0.69019224, 0.68809361, 0.55854128,
0.55394177, 0.53829688, 0.57263777, 0.55523789, 0.68956156,
0.56765985, 0.55854128, 0.56268193, 0.56573304, 0.53466077,
0.53829688, 0.5540679 , 0.55699288, 0.6761762 ])
```

In [802…
```
model.predict([[40, 1, 0 , 2]])
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, b
ut LinearRegression was fitted with feature names
  warnings.warn(
```

Out[802…
```
array([2.2719247])
```

In [ ]:
```
```