Importing the libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Data Collection & Analysis

In [2]:

```
df=pd.read_csv("C:/Users/Asus/Downloads/insurance.csv")
df
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

In [3]:

```
df.shape
```

Out[3]:

```
(1338, 7)
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

Categorical Features:

- Sex
- Smoker
- Region

Data Analysis

In [5]:

```
df.describe()
```

Out[5]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [7]:

```
df.nunique()
```

Out[7]:

```
age           47
sex            2
bmi          548
children       6
smoker         2
region         4
charges     1337
dtype: int64
```

In [8]:

```
df.columns
```

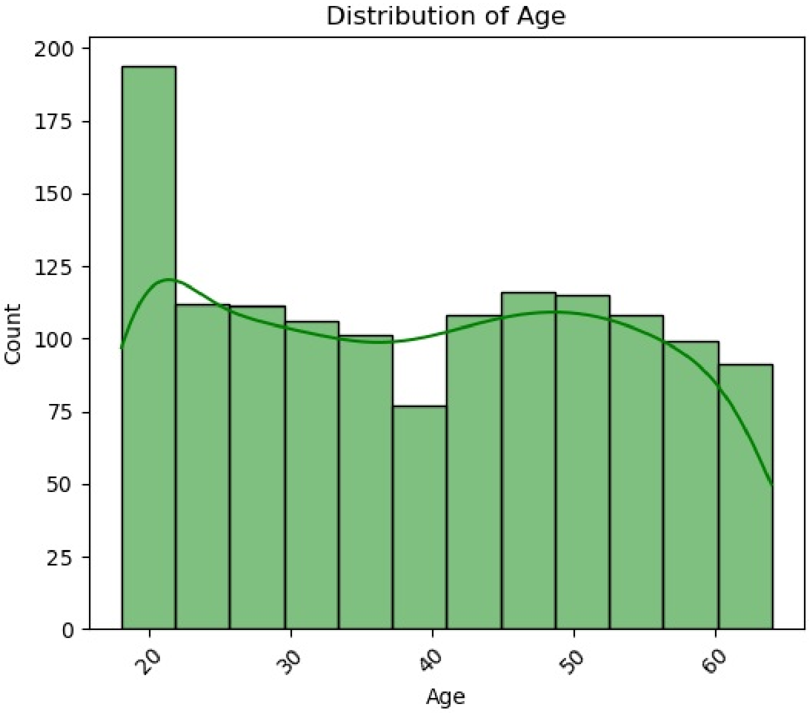
Out[8]:

```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

Data visualization

In [9]:

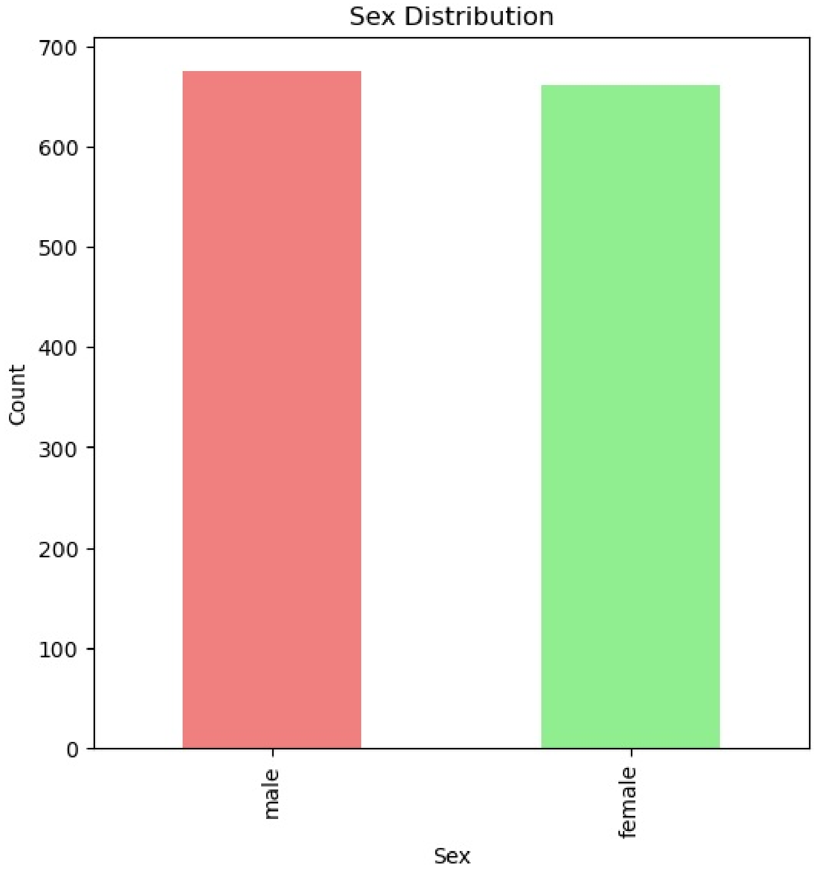
```
plt.figure(figsize=(6, 5))
sns.histplot(df['age'], kde=True, color='green', edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Distribution of Age')
plt.xticks(rotation=45)
plt.show()
```



In [10]:

```
# Gender column using value_counts and bar plot
plt.figure(figsize=(6, 6))
df['sex'].value_counts().plot(kind='bar', color=['lightcoral', 'lightgreen'])
```

```
plt.title('Sex Distribution')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()
```



In [11]:

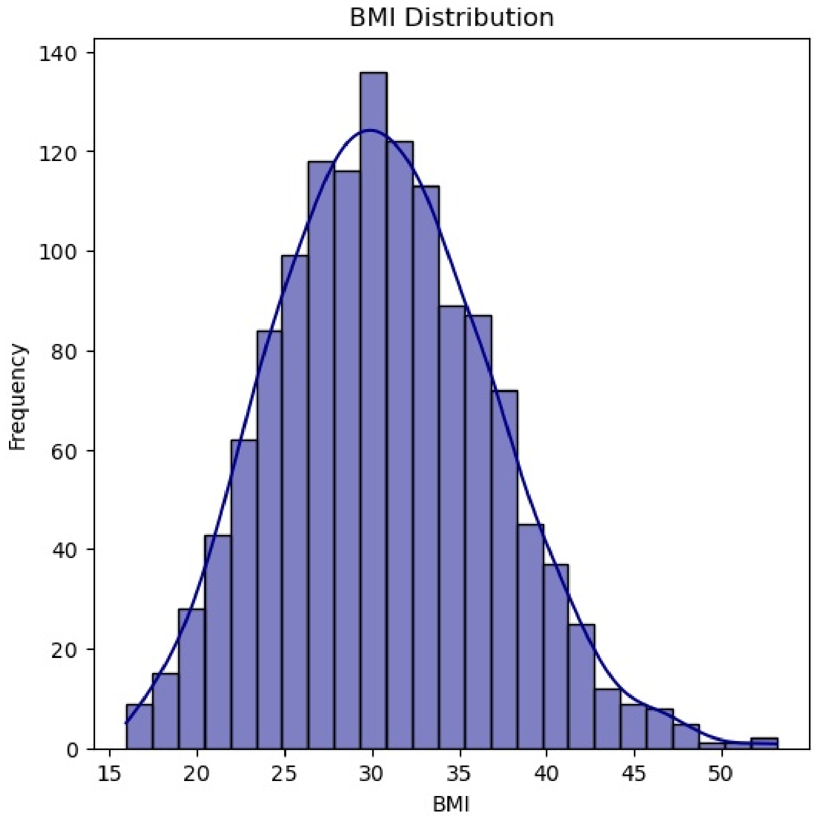
```
df['sex'].value_counts()
```

Out[11]:

```
male      676
female    662
Name: sex, dtype: int64
```

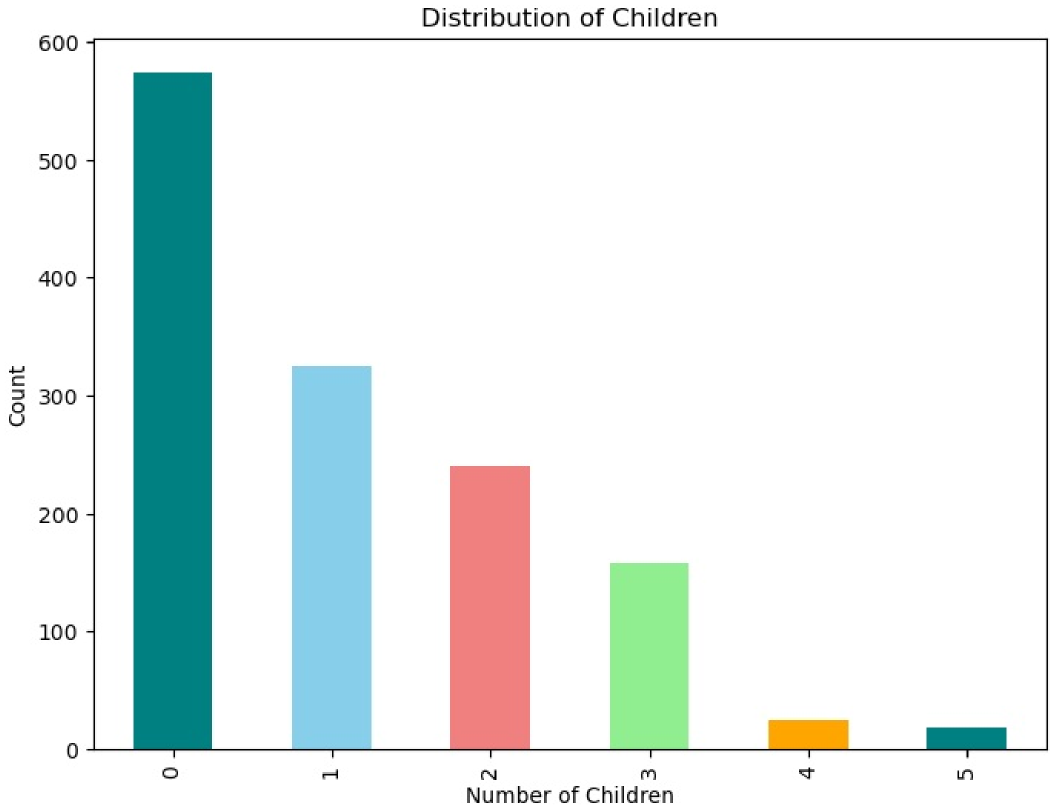
In [12]:

```
# BMI distribution using histplot with a dark color
plt.figure(figsize=(6, 6))
sns.histplot(df['bmi'], kde=True, color='darkblue')
plt.title('BMI Distribution')
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.show()
```



Normal BMI Range --> 18.5 to 24.9

In [14]:
```
plt.figure(figsize=(8, 6))
colors = ['teal', 'skyblue', 'lightcoral', 'lightgreen', 'orange']
df['children'].value_counts().sort_index().plot(kind='bar', color=colors)
plt.title('Distribution of Children')
plt.xlabel('Number of Children')
plt.ylabel('Count')
plt.show()
```



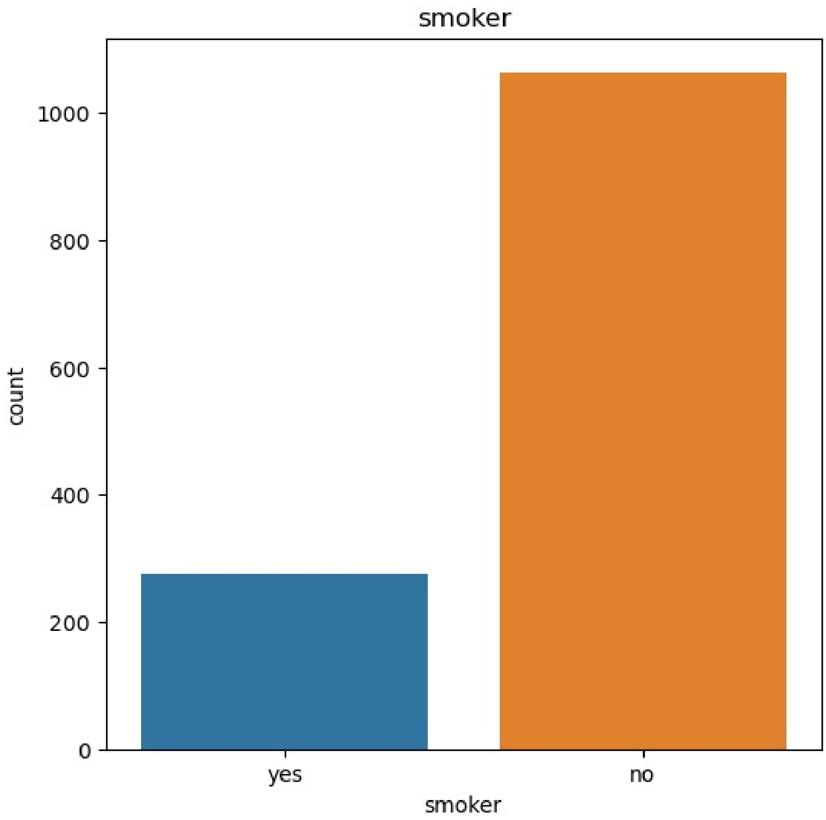
In [15]:
```
df['children'].value_counts()
```

Out[15]:

```
0    574
1    324
2    240
3    157
4     25
5     18
Name: children, dtype: int64
```

In [16]:

```
# smoker column
plt.figure(figsize=(6,6))
sns.countplot(x='smoker', data=df)
plt.title('smoker')
plt.show()
```



In [17]:

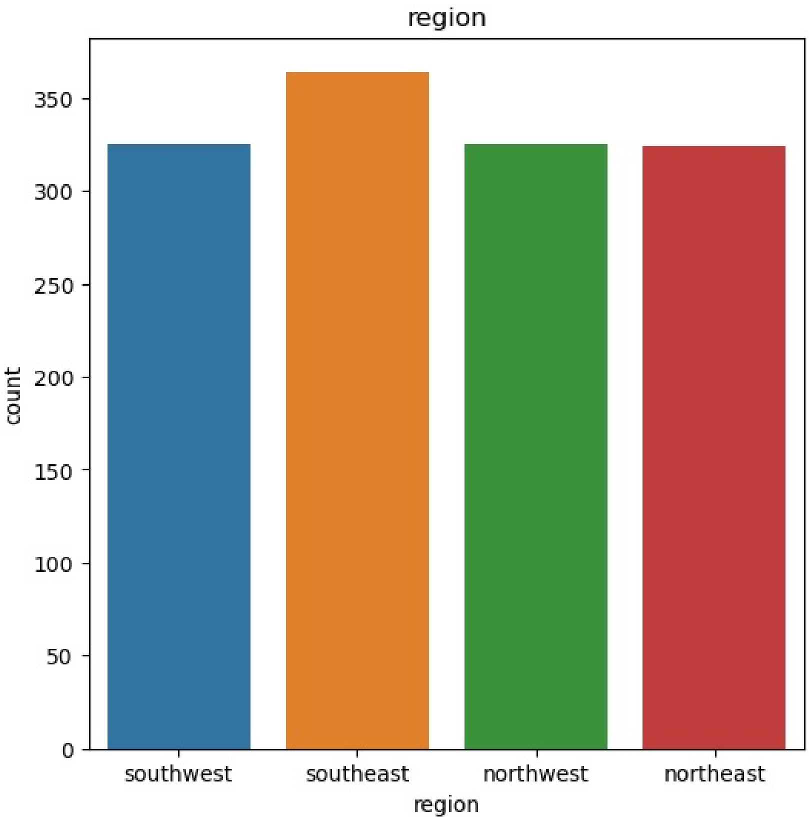
```
df['smoker'].value_counts()
```

Out[17]:

```
no     1064
yes     274
Name: smoker, dtype: int64
```

In [18]:

```
# region column
plt.figure(figsize=(6,6))
sns.countplot(x='region', data=df)
plt.title('region')
plt.show()
```



In [19]:

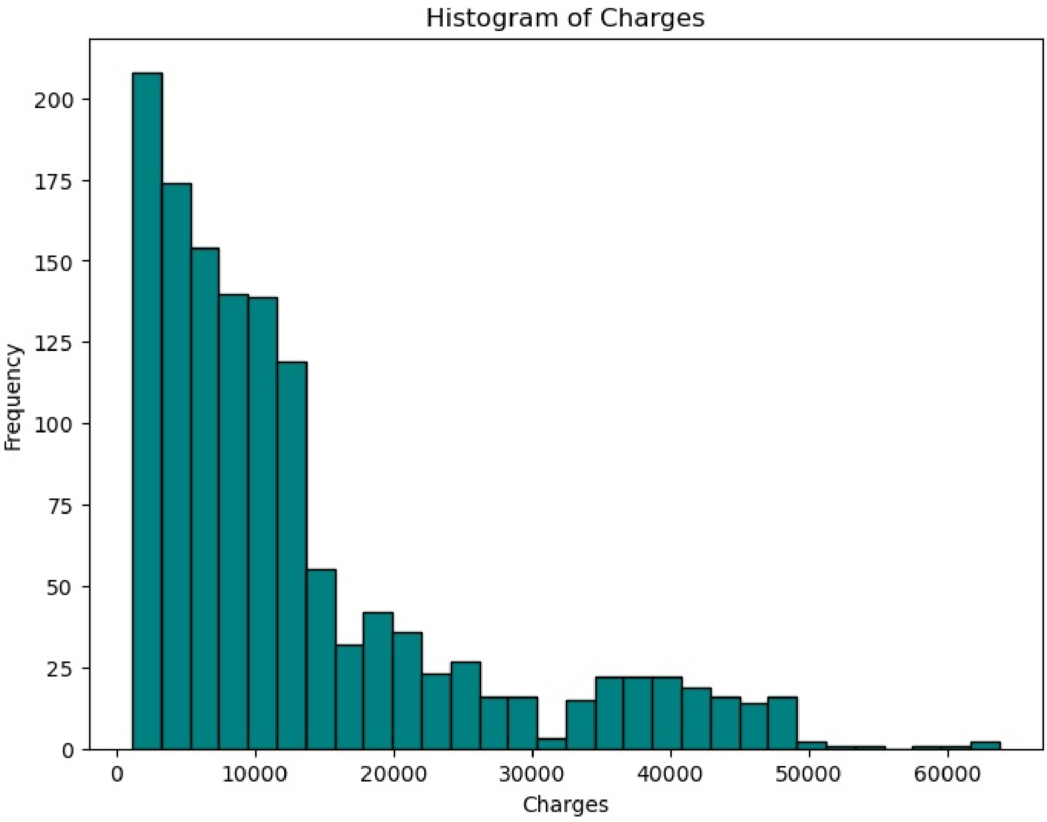
```
df['region'].value_counts()
```

Out[19]:

```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [20]:

```
plt.figure(figsize=(8, 6))
plt.hist(df['charges'], bins=30, color='teal', edgecolor='black')
plt.title('Histogram of Charges')
plt.xlabel('Charges')
plt.ylabel('Frequency')
plt.show()
```



Data Pre-Processing

Encoding the categorical features

In [21]:

```
# encoding sex column
df.replace({'sex':{'male':0,'female':1}}, inplace=True)

 # encoding 'smoker' column
df.replace({'smoker':{'yes':0,'no':1}}, inplace=True)

# encoding 'region' column
df.replace({'region':{'southeast':0,'southwest':1,'northeast':2,'northwest':3}}, inplace=True)
```

Splitting the Features and Target

In [22]:

```
X = df.drop(columns='charges', axis=1)
Y = df['charges']
```

In [23]:

```
print(X)
```

```
      age  sex     bmi  children  smoker  region
0      19    1  27.900         0       0       1
1      18    0  33.770         1       1       0
2      28    0  33.000         3       1       0
3      33    0  22.705         0       1       3
4      32    0  28.880         0       1       3
...   ...  ...     ...       ...     ...     ...
1333   50    0  30.970         3       1       3
1334   18    1  31.920         0       1       2
1335   18    1  36.850         0       1       0
1336   21    1  25.800         0       1       1
1337   61    1  29.070         0       0       3

[1338 rows x 6 columns]
```

In [24]:

```
print(Y)
```

```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

Splitting the data into Training data & Testing Data

In [25]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

In [26]:

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(1338, 6) (1070, 6) (268, 6)
```

Model Training

Linear Regression

In [27]:

```
# loading the Linear Regression model
regressor = LinearRegression()
```

In [28]:

```
regressor.fit(X_train, Y_train)
```

Out[28]:

```
LinearRegression()
```

Model Evaluation

In [29]:

```
# prediction on training data
training_data_prediction =regressor.predict(X_train)
```

In [30]:

```
# R squared value
r2_train = metrics.r2_score(Y_train, training_data_prediction)
```

```
print('R squared value : ', r2_train)
```

```
R squared value :  0.751505643411174
```

In [31]:

```
# prediction on test data
test_data_prediction =regressor.predict(X_test)
```

In [32]:

```
# R squared value
r2_test = metrics.r2_score(Y_test, test_data_prediction)
print('R squared value : ', r2_test)
```

```
R squared value :  0.7447273869684077
```

Building a Predictive System

In [33]:

```
input_data = (31,1,25.74,0,1,0)
```

In [34]:

```
# changing input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)
```

In [35]:

```
# reshape the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```

In [36]:

```
prediction = regressor.predict(input_data_reshaped)
print(prediction)
```

```
[3760.0805765]
```

```
C:\Users\Asus\anacondar\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names
, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [37]:

```
print('The insurance cost is USD ', prediction[0])
```

```
The insurance cost is USD  3760.0805764960496
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js