In [1]:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:

```
# Importing the dataset
dataset = pd.read_csv(r"C:\Users\SSD\Downloads\health cost.csv")
```

In [3]:

```
dataset
```

	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 [4]:

```
dataset.head()
```

	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

Number of Records&Columns

In [5]:

```
dataset.shape
```

```
(1338, 7)
```

In [6]:

```
dataset.tail()
```

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	no	northwest	10600.5483
1334	18	female	31.92	0	no	northeast	2205.9808
1335	18	female	36.85	0	no	southeast	1629.8335
1336	21	female	25.80	0	no	southwest	2007.9450
1337	61	female	29.07	0	yes	northwest	29141.3603

List of Columns:

In [7]:

```
dataset.columns
```

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

Columns Datatype:

In [8]:

```
dataset. dtypes
```

```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

Data Information:

In [9]:

```
dataset .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
```

Check for Duplicate Records:

In [10]:
```
dataset.duplicated().any()
```

```
True
```

In [11]:
```
dataset[dataset.duplicated()]
```

	age	sex	bmi	children	smoker	region	charges
581	19	male	30.59	0	no	northwest	1639.5631

In [12]:
```
dataset=dataset.drop_duplicates()
```

In [13]:
```
dataset.shape
```

```
(1337, 7)
```

Checking for missing values:

In [14]:
```
dataset .isnull().sum()
```

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

In [15]:
```
dataset .isnull().any()
```

```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

List of Categorical and Numeric Columns:

In [16]:

```
Numerical = ["age", "bmi", "children", "charges"]
Categorical = ["sex", "smoker", "region"]

print('Numerical: ', ', '.join(Numerical))
print('Categorical: ', ', '.join(Categorical))
```

```
Numerical:  age, bmi, children, charges
Categorical:  sex, smoker, region
```

Statistcal Measure of Numeric Columns:

In [17]:

```
dataset.describe()
```

	age	bmi	children	charges
count	1337.000000	1337.000000	1337.000000	1337.000000
mean	39.222139	30.663452	1.095737	13279.121487
std	14.044333	6.100468	1.205571	12110.359656
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.290000	0.000000	4746.344000
50%	39.000000	30.400000	1.000000	9386.161300
75%	51.000000	34.700000	2.000000	16657.717450
max	64.000000	53.130000	5.000000	63770.428010

In [18]:

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

```
male      675
female    662
Name: sex, dtype: int64
```

In [19]:

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

```
no     1063
yes     274
Name: smoker, dtype: int64
```

In [20]:

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

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

In [21]:

```
dataset['region'].unique()
```

```
array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

In [22]:

```
dataset['region'].nunique()
```

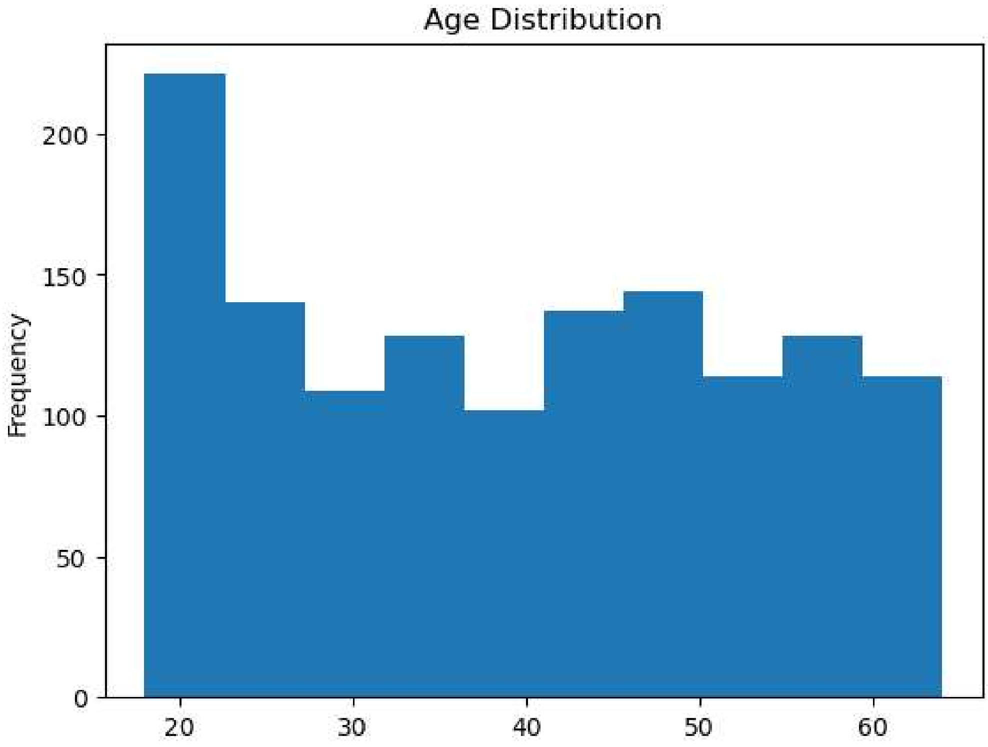
```
4
```

Data Visulization:

Histogram:Numeric Columns:

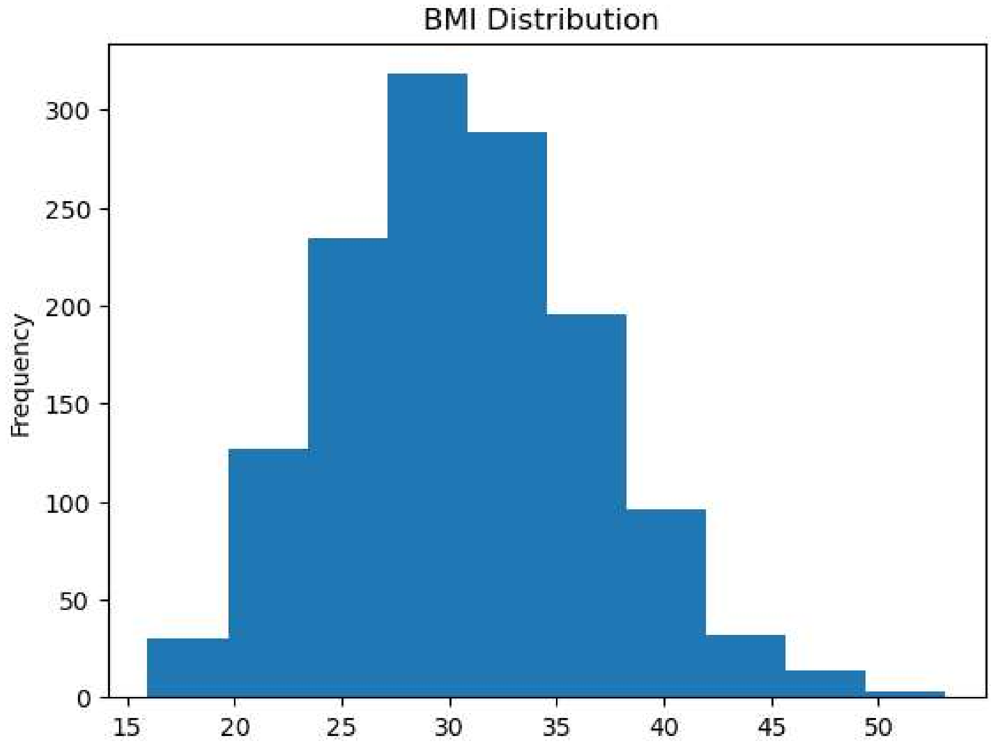
In [23]:

```
dataset['age'].plot(kind = 'hist')
plt.title("Age Distribution")
plt.show()
```



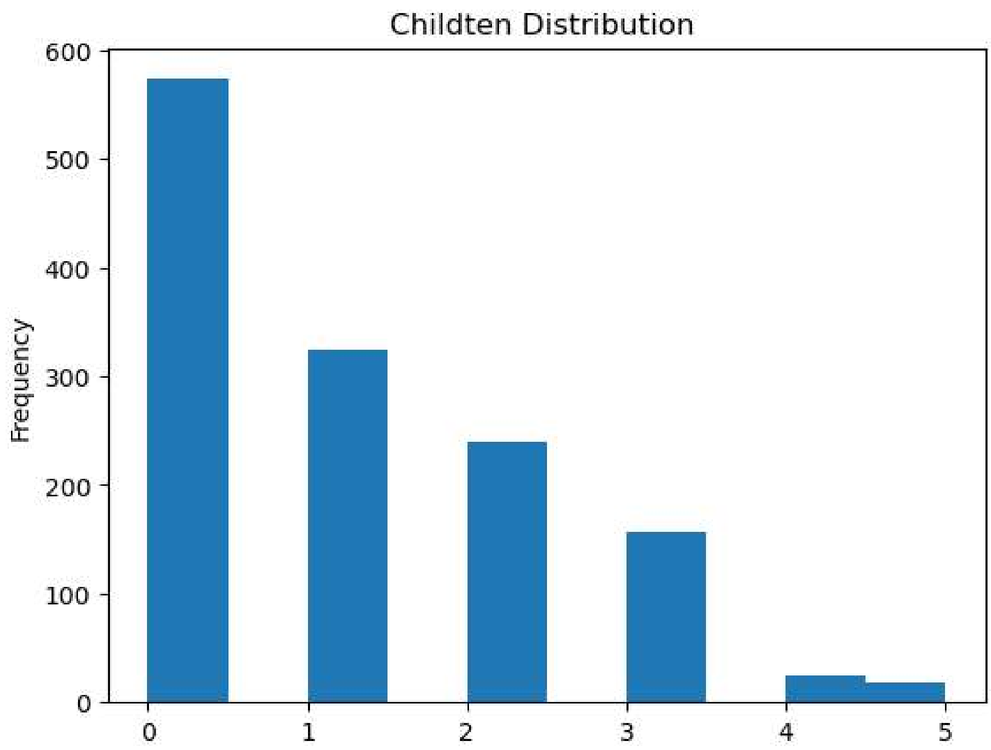
In [24]:

```
dataset['bmi'].plot(kind = 'hist')
plt.title("BMI Distribution ")
plt.show()
```



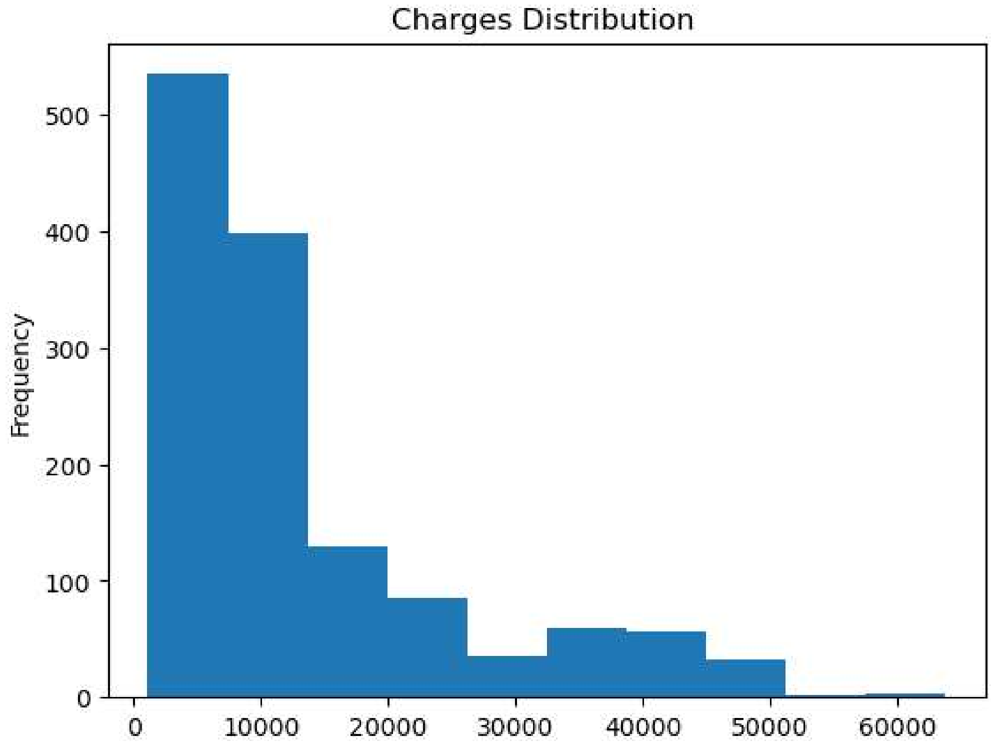
In [25]:

```
dataset['children'].plot(kind = 'hist')
plt.title("Childten Distribution ")
plt.show()
```



In [26]:

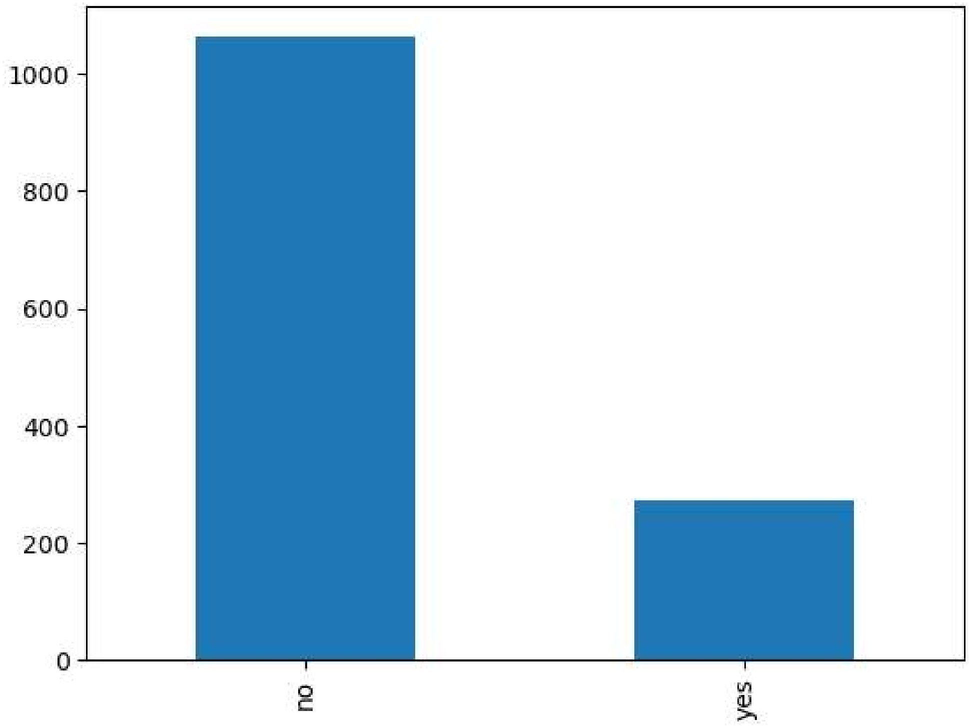
```
dataset['charges'].plot(kind = 'hist')
plt.title("Charges Distribution ")
plt.show()
```



Bar Graph-Categorical Columns:

```
In [27]: dataset['smoker'].value_counts().plot(kind= 'bar')
```

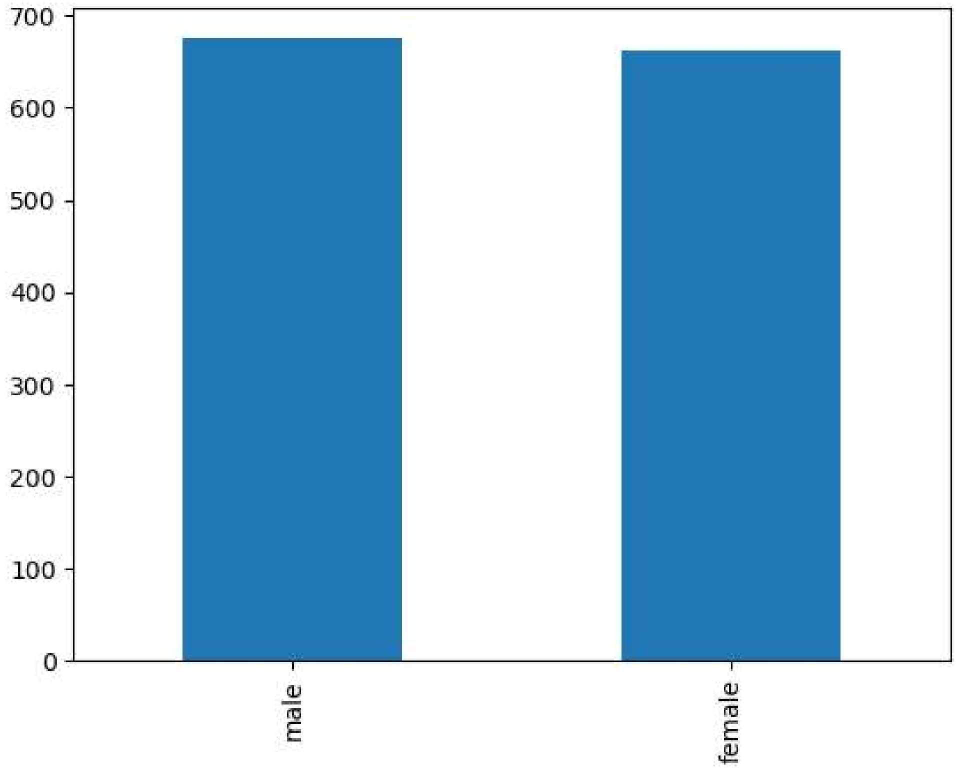
```
<Axes: >
```



In [28]:

```
dataset['sex'].value_counts().plot(kind= 'bar')
```

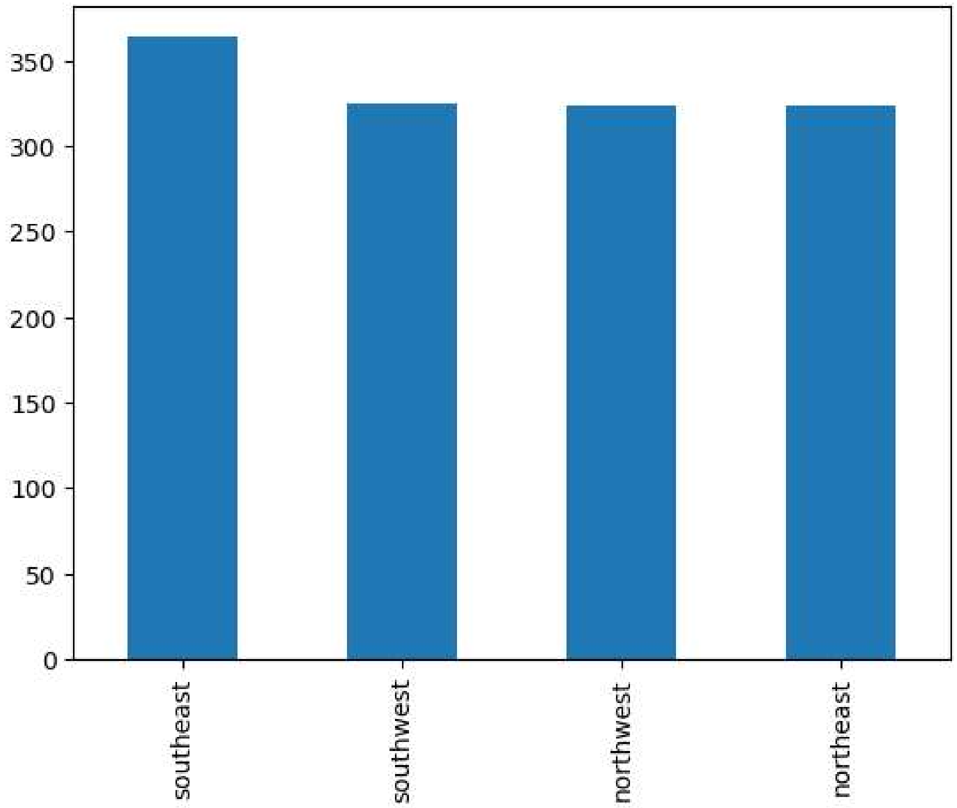
```
<Axes: >
```



In [29]:

```
dataset['region'].value_counts().plot(kind= 'bar')
```

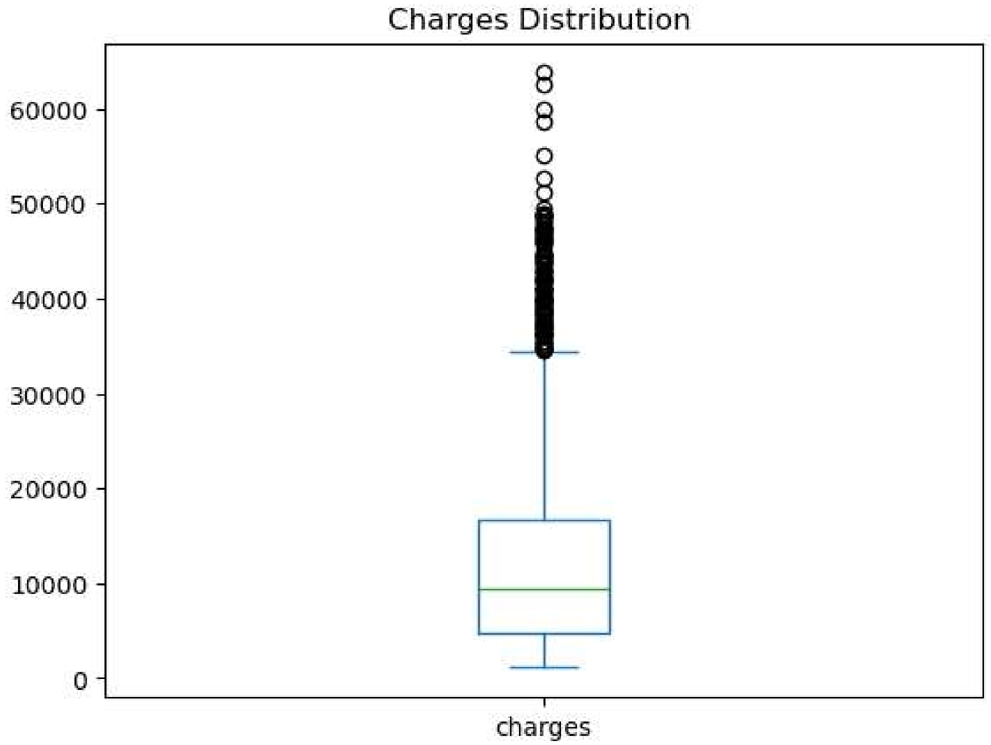
```
<Axes: >
```



Box plot-Numeric Columns:

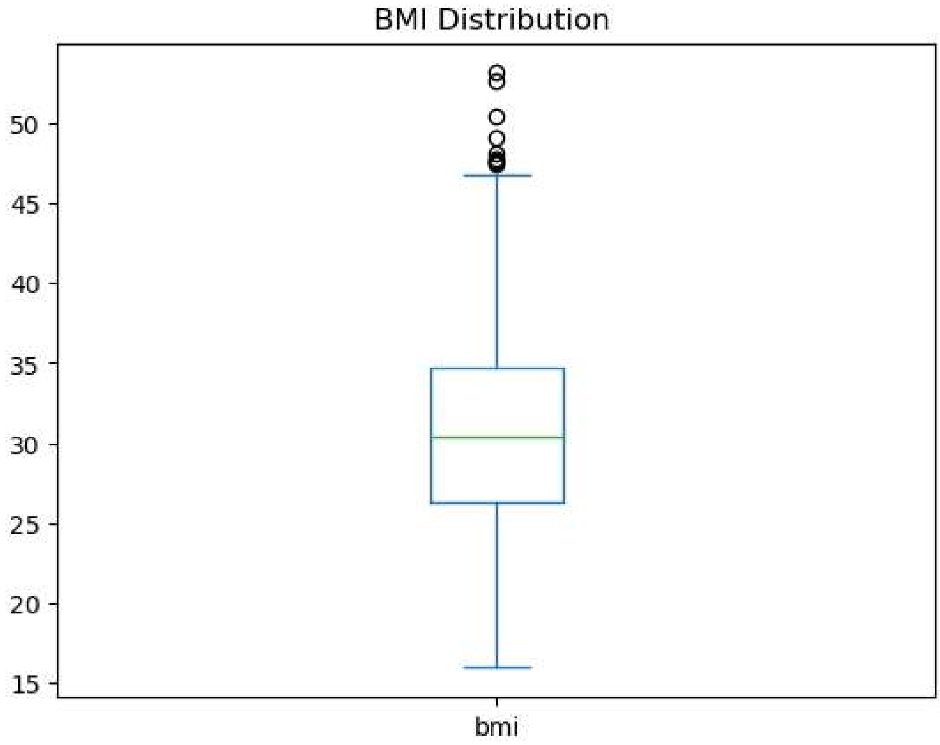
In [30]:

```
dataset['charges'].plot(kind = 'box')
plt.title("Charges Distribution ")
plt.show()
```



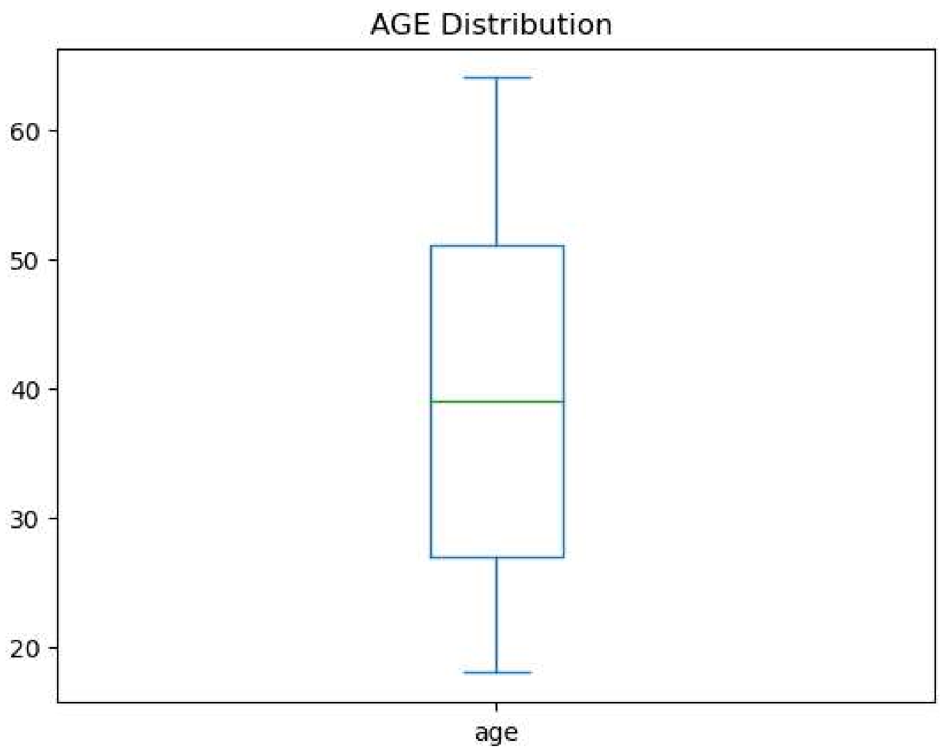
In [31]:

```
dataset['bmi'].plot(kind = 'box')
plt.title("BMI Distribution ")
plt.show()
```



In [32]:

```
dataset['age'].plot(kind = 'box')
plt.title("AGE Distribution ")
plt.show()
```



Pie Chart-Region

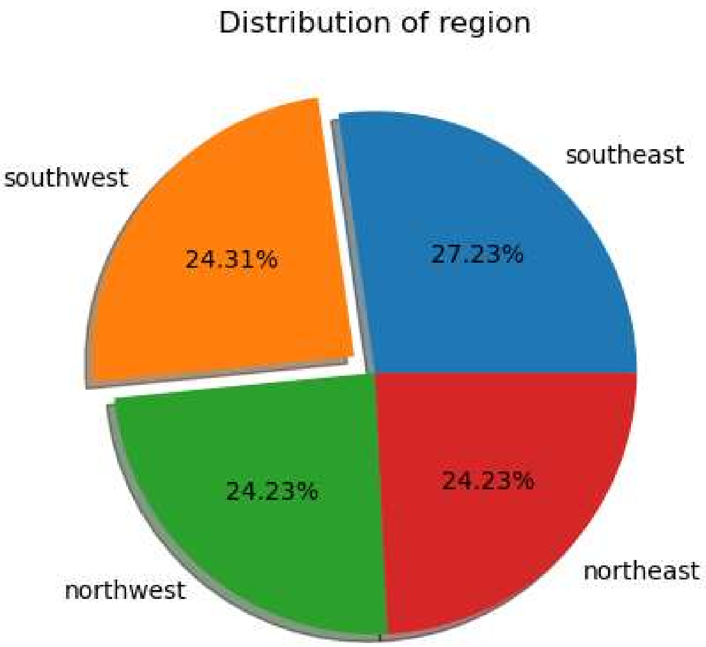
In [33]:

```
region_count = dataset['region'].value_counts()
region_count
```

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

In [34]:

```
plt.pie(labels=region_count.index,
          x=region_count.values,
          shadow=True,
          autopct = '%.2f%%',
          explode = (0,0.1,0,0))
plt.title("Distribution of region")
plt.show()
```



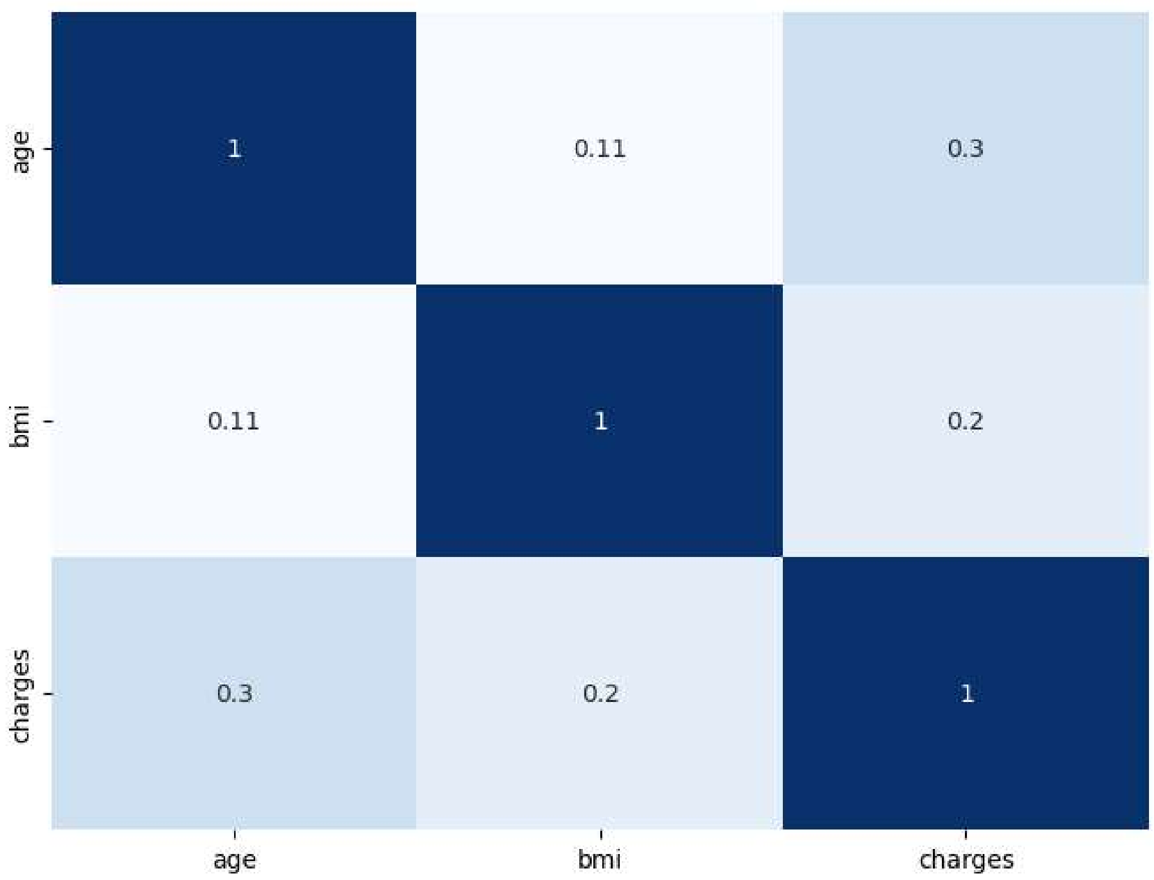
Correlation Matrix:

In [35]:

```
import seaborn as sns
```

In [36]:

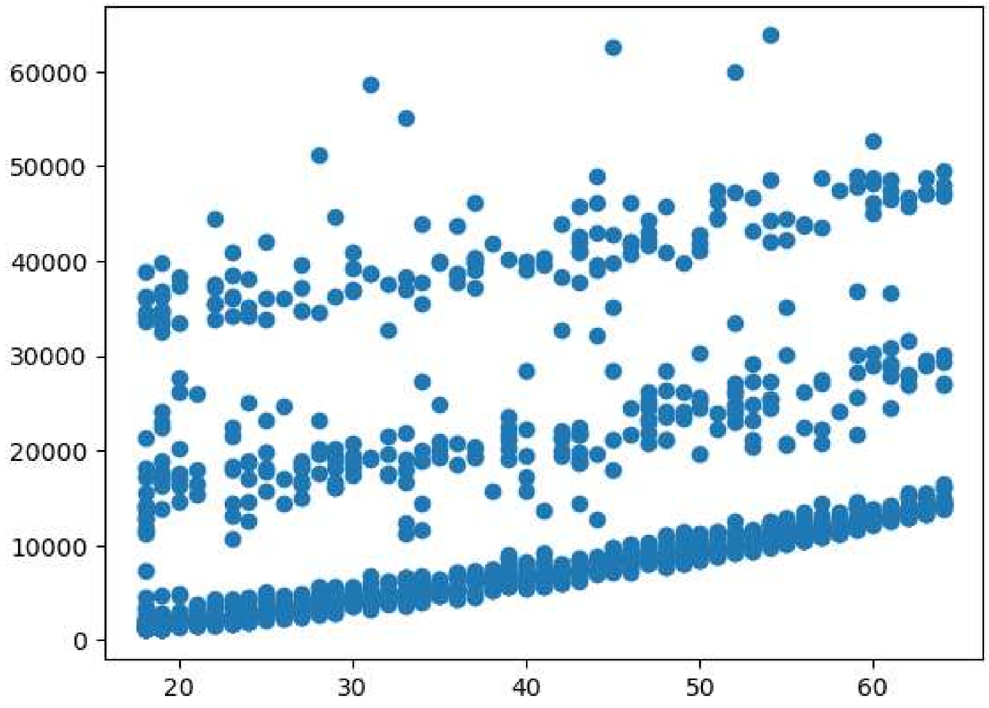
```
corr_data = dataset[["age","bmi","charges"]].corr()
plt.figure(figsize=(8,6))
sns.heatmap(round(corr_data,2),annot=True, cmap="Blues", cbar=False)
plt.show()
```



Scatter plot:

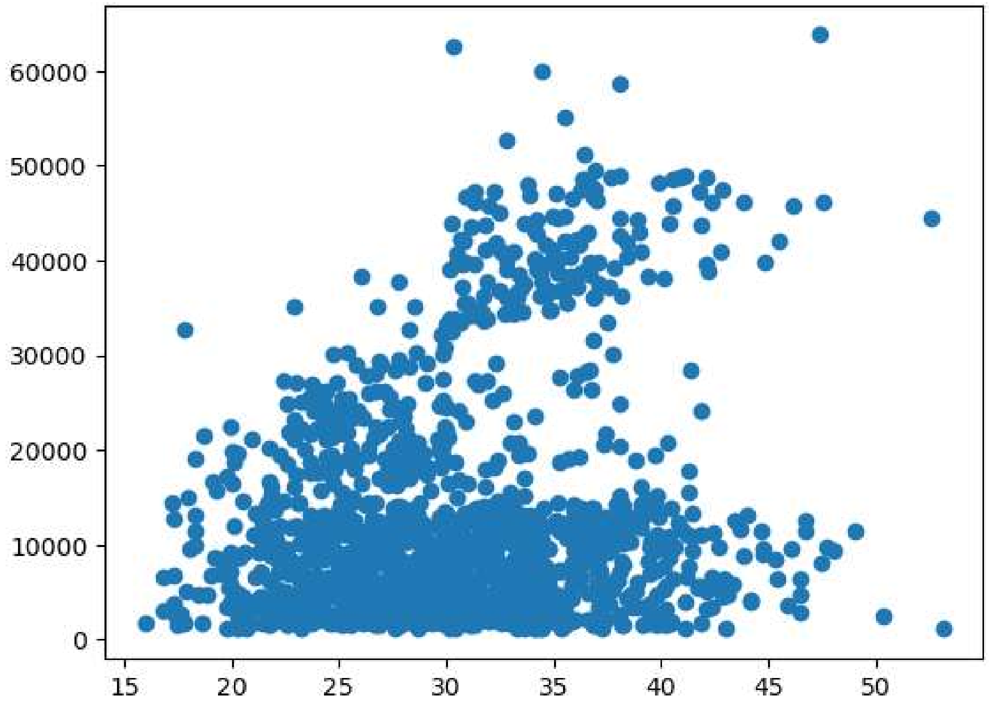
```
In [37]: plt.scatter(data = dataset, x = 'age', y = 'charges')
```

```
<matplotlib.collections.PathCollection at 0x2bf38b0fad0>
```



```
In [38]: plt.scatter(data = dataset, x = 'bmi', y = 'charges')
```

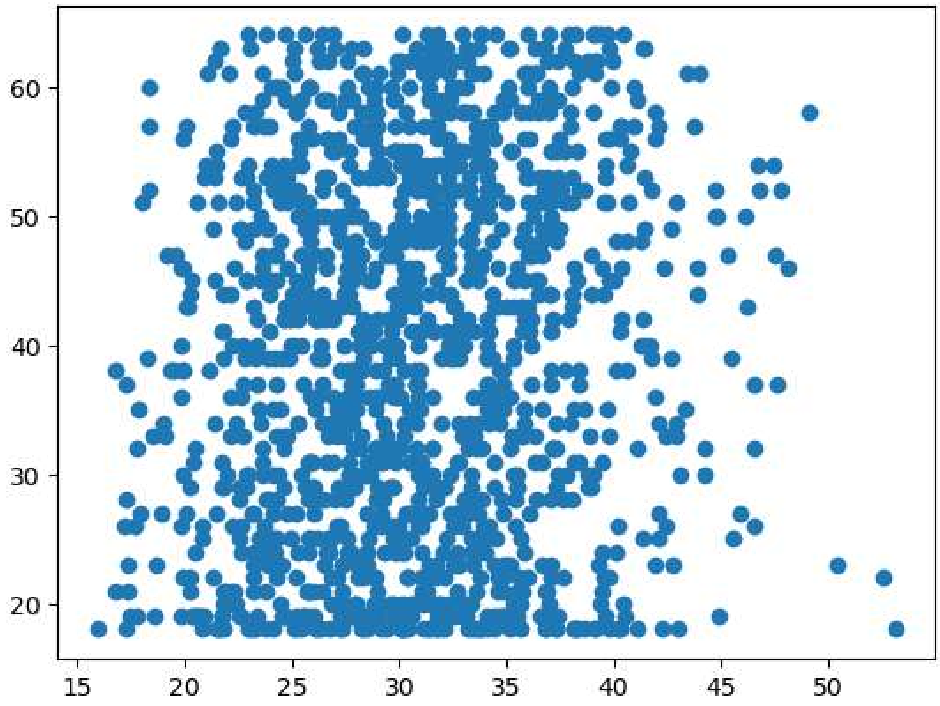
```
<matplotlib.collections.PathCollection at 0x2bf38b57bd0>
```



In [39]:

```
plt.scatter(data = dataset, x = 'bmi', y = 'age')
```

```
<matplotlib.collections.PathCollection at 0x2bf38d0f790>
```



Charges Cost for Smoker and non smoker:

In [40]:

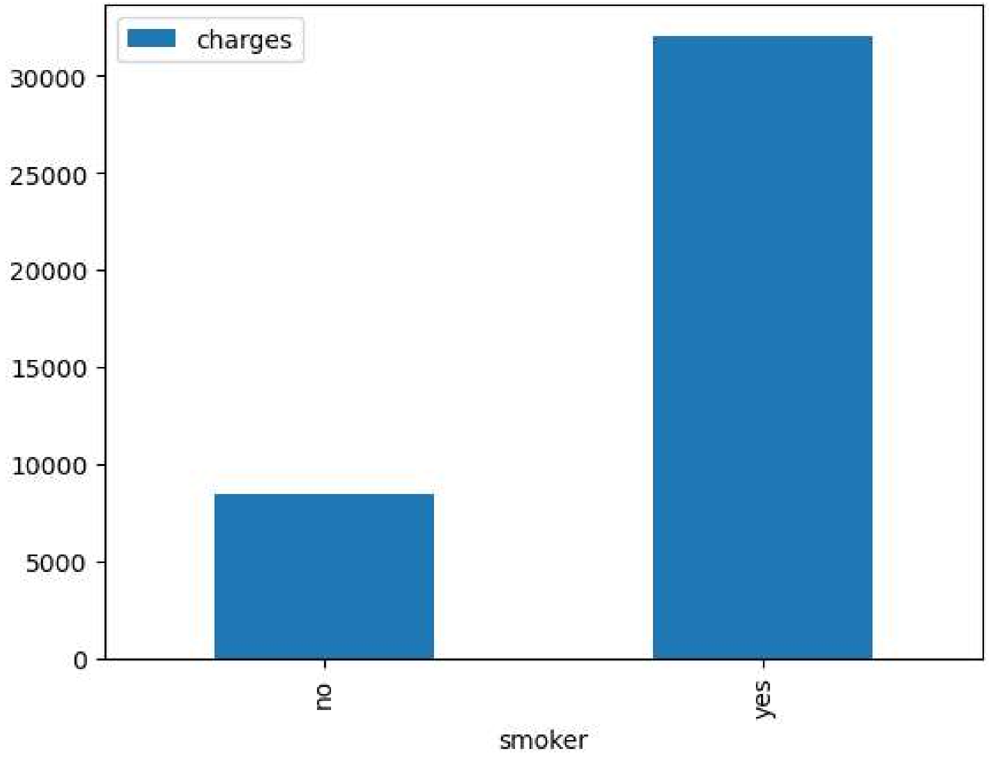
```
smoker_df = dataset.groupby("smoker")["charges"].mean().reset_index()
smoker_df
```

	smoker	charges
0	no	8440.660307
1	yes	32050.231832

In [41]:

```
smoker_df .plot(kind = 'bar', x = 'smoker', y = 'charges')
```

```
<Axes: xlabel='smoker'>
```



Charges Cost for Male and Female:

In [42]:

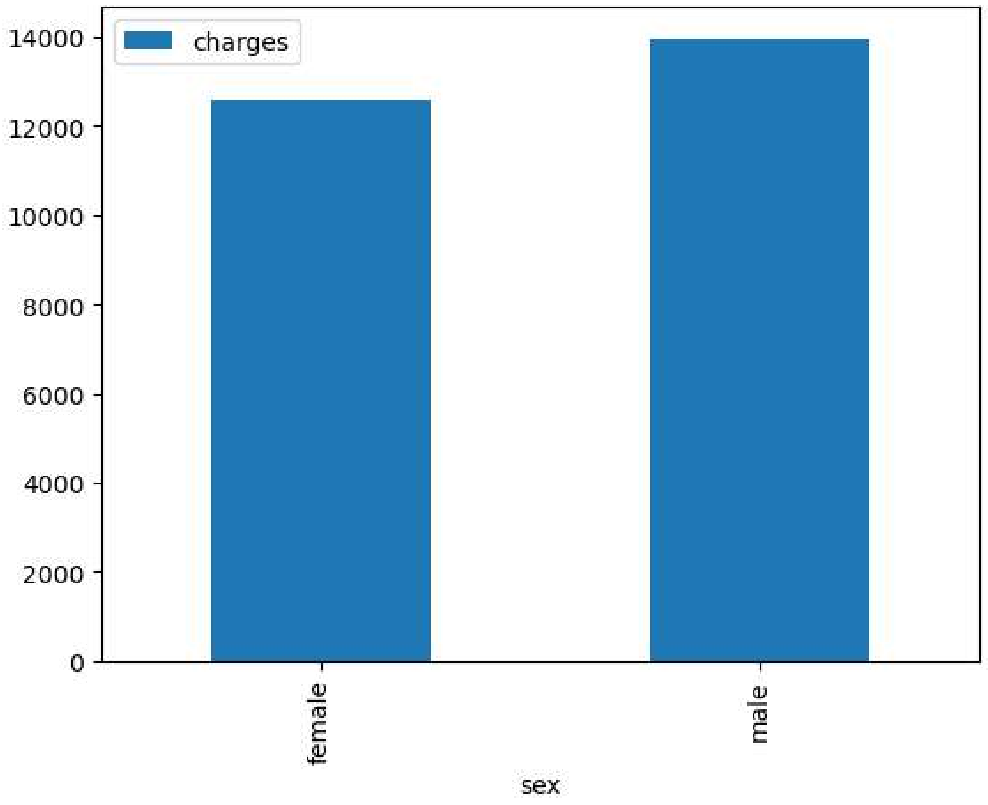
```
gender_df = dataset.groupby("sex")["charges"].mean().reset_index()
gender_df
```

	sex	charges
0	female	12569.578844
1	male	13974.998864

In [43]:

```
gender_df .plot(kind = 'bar', x = 'sex', y = 'charges')
```

```
<Axes: xlabel='sex'>
```



Charges Cost for Region Wise

In [44]:

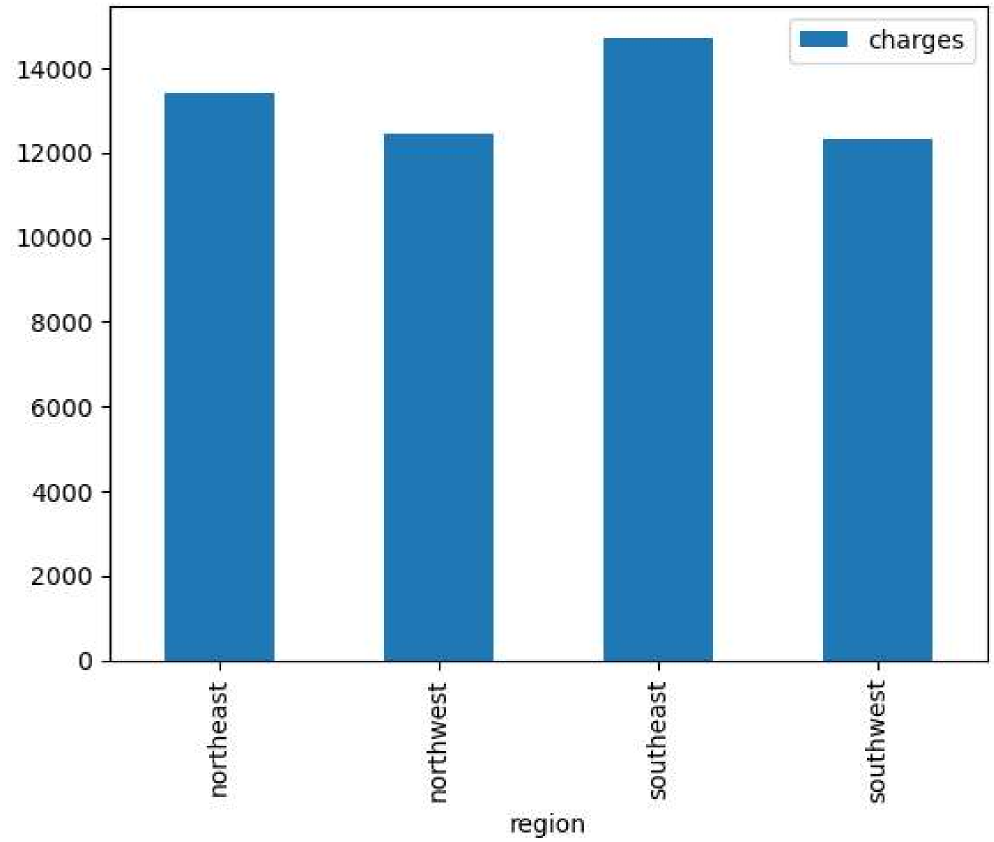
```
region_df = dataset.groupby("region")["charges"].mean().reset_index()
region_df
```

	region	charges
0	northeast	13406.384516
1	northwest	12450.840844
2	southeast	14735.411438
3	southwest	12346.937377

In [45]:

```
region_df .plot(kind = 'bar', x = 'region', y = 'charges')
```

```
<Axes: xlabel='region'>
```



In [46]:

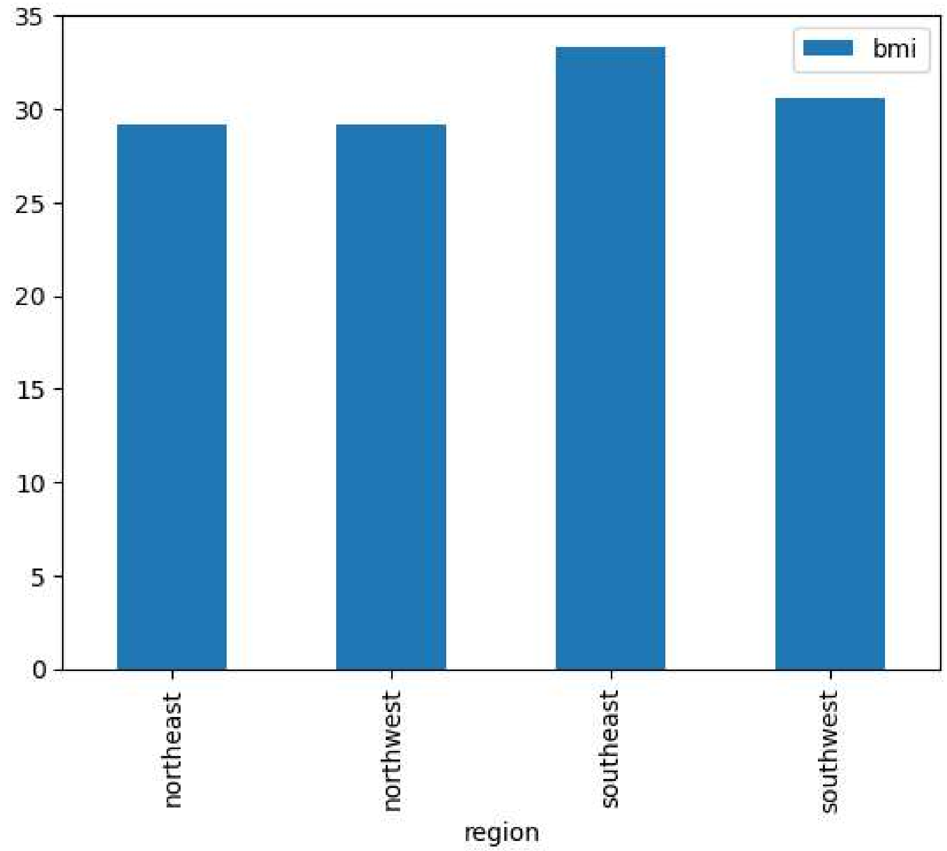
```
region_bmi_df = dataset.groupby("region")["bmi"].mean().reset_index()
region_bmi_df
```

	region	bmi
0	northeast	29.173503
1	northwest	29.195494
2	southeast	33.355989
3	southwest	30.596615

In [47]:

```python
region_bmi_df .plot(kind = 'bar', x = 'region', y = 'bmi')
```

```
<Axes: xlabel='region'>
```



Machine Learning Model Development:

In [48]:

```python
dataset.head()
```

	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

Label Encoding:

In [49]:

```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [50]:

```
dataset['sex'] = le.fit_transform(dataset['sex'])
dataset['smoker'] = le.fit_transform(dataset['smoker'])
dataset['region'] = le.fit_transform(dataset['region'])
```

```
C:\Users\SSD\AppData\Local\Temp\ipykernel_9640\2119898308.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
ta.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  dataset['sex'] = le.fit_transform(dataset['sex'])
C:\Users\SSD\AppData\Local\Temp\ipykernel_9640\2119898308.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
ta.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  dataset['smoker'] = le.fit_transform(dataset['smoker'])
C:\Users\SSD\AppData\Local\Temp\ipykernel_9640\2119898308.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
ta.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  dataset['region'] = le.fit_transform(dataset['region'])
```

In [51]:

```
dataset.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520

split the dataset in feature variable and terget variable

In [52]:

```
X = dataset.drop(columns=['charges'], axis=1)
y = dataset['charges']
```

In [53]:
```
X
```

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

```
1337 rows × 6 columns
```

In [54]:
```
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: 1337, dtype: float64
```

Split the dataset into traing and testing

In [55]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.85,random_state=0)
```

In [56]:
```
X_train.shape
```

```
(200, 6)
```

In [57]:
```
X_test.shape
```

```
(1137, 6)
```

In [58]:
```
y_train.shape
```

```
(200,)
```

In [59]:
```
y_test.shape
```

```
(1137,)
```

Feature Scaling

In [60]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train
```

```
array([[-1.49354533,  0.94169658, -0.094995  , -0.88443957, -0.56195149,
         0.40011013],
       [ 1.03788743, -1.06191317,  2.58196784,  0.78431434, -0.56195149,
         1.29923403],
       [ 1.24884016,  0.94169658, -2.00330094, -0.88443957, -0.56195149,
        -1.39813765],
       ...,
       [-0.72005198,  0.94169658,  1.06081927,  0.78431434, -0.56195149,
         0.40011013],
       [-1.42322775,  0.94169658,  0.77631114, -0.88443957, -0.56195149,
        -0.49901376],
       [ 0.96756985,  0.94169658, -0.69795825,  0.78431434, -0.56195149,
        -1.39813765]])
```

In [61]:
```
X_test
```

```
array([[-1.49354533, -1.06191317,  1.46979971, -0.88443957, -0.56195149,
         0.40011013],
       [ 0.54566439, -1.06191317, -0.21946731, -0.05006262, -0.56195149,
         0.40011013],
       [ 0.68629955,  0.94169658,  0.10060434, -0.05006262, -0.56195149,
        -1.39813765],
       ...,
       [ 1.60042804, -1.06191317, -0.92588806, -0.88443957, -0.56195149,
         1.29923403],
       [ 0.54566439,  0.94169658, -0.8596106 , -0.05006262,  1.77951304,
         0.40011013],
       [ 1.31915774,  0.94169658,  0.20729489, -0.05006262, -0.56195149,
         0.40011013]])
```

In [62]:

```
from sklearn.linear_model import LinearRegression
linear_reg_model = LinearRegression()
linear_reg_model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

In [63]:

```
y_pred = linear_reg_model.predict(X_test)
y_pred
```

```
array([ 5964.93213623,  9400.23586776, 10980.10911736, ...,
       10374.96718278, 33084.62554112, 12293.47788549])
```

In [64]:

```
bias = linear_reg_model .score(X_train,y_train)
bias
```

```
0.7667435043084556
```

In [65]:

```
variance = linear_reg_model .score(X_test,y_test)
variance
```

```
0.7409918864948746
```

In [66]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = linear_reg_model , X = X_train, y = y_train, cv = 10
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 73.83 %
Standard Deviation: 9.72 %
```

In [67]:

```
from sklearn.neighbors import KNeighborsRegressor
Knn_reg_model=KNeighborsRegressor()
Knn_reg_model.fit(X,y)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

In [68]:

```
y_pred =Knn_reg_model.predict(X_test)
y_pred
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but KNe
s fitted with feature names
  warnings.warn(
```

```
array([3920.45474, 3920.45474, 4208.16651, ..., 3920.45474, 3920.45474,
       3920.45474])
```

In [69]:

```
bias = Knn_reg_model .score(X_train,y_train)
bias
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but KNe
s fitted with feature names
  warnings.warn(
```

```
-0.6133027610766579
```

In [70]:

```
variance =Knn_reg_model .score(X_test,y_test)
variance
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but KNe
s fitted with feature names
  warnings.warn(
```

```
-0.5866890802537086
```

In [71]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = Knn_reg_model , X = X_train, y = y_train, cv = 8)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 78.69 %
Standard Deviation: 8.07 %
```

In [84]:

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X, y)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegres
sor()
```

In [85]:

```
y_pred =regressor.predict(X_test)
y_pred
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Dec
was fitted with feature names
  warnings.warn(
```

```
array([ 2196.4732 ,  2196.4732 ,  1694.7964 , ...,  2117.33885,
       12829.4551 ,  1694.7964 ])
```

In [74]:

```
bias =  regressor.score(X_train,y_train)
bias
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Dec
was fitted with feature names
  warnings.warn(
```

```
0.02669476985969188
```

In [75]:

```
variance = regressor.score(X_test,y_test)
variance
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Dec
was fitted with feature names
  warnings.warn(
```

```
-0.039977766177715424
```

In [76]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor , X = X_train, y = y_train, cv = 5)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 74.39 %
Standard Deviation: 7.54 %
```

In [77]:

```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
regressor.fit(X, y)
```

```
▾ RandomForestRegressor
RandomForestRegres
sor()
```

In [78]:

```
y_pred =regressor.predict(X_test)
y_pred
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Ran
was fitted with feature names
  warnings.warn(
```

```
array([ 2240.25688988,  2240.25688988,  1717.50859   , ...,
        2048.83020169, 15238.6465423 ,  1717.50859   ])
```

In [79]:

```
bias =  regressor.score(X_train,y_train)
bias
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Ran
was fitted with feature names
  warnings.warn(
```

```
0.11018301336085612
```

In [86]:

```
variance = regressor.score(X_test,y_test)
variance
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Dec
was fitted with feature names
  warnings.warn(
```

```
-0.04188776835209196
```

In [87]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor , X = X_train, y = y_train, cv = 5)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 77.68 %
Standard Deviation: 10.07 %
```

In [89]:

```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor(n_estimators=30)
regressor.fit(X, y)
```

```
▾ RandomForestRegressor
RandomForestRegressor(n_estima
tors=30)
```

In [90]:

```
variance = regressor.score(X_test,y_test)
variance
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Ran
was fitted with feature names
  warnings.warn(
```

```
0.010205389515185237
```

In [91]:

```
bias =  regressor.score(X_train,y_train)
bias
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but Ran
was fitted with feature names
  warnings.warn(
```

```
0.048764605345108514
```

In [92]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor , X = X_train, y = y_train, cv = 5)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 85.13 %
Standard Deviation: 4.76 %
```

In [93]:

```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor(n_estimators=50)
regressor.fit(X, y)
```

```
▾ RandomForestRegressor
RandomForestRegressor(n_estima
tors=50)
```

In [94]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor , X = X_train, y = y_train, cv = 5)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 84.92 %
Standard Deviation: 4.40 %
```

In [95]:

```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor(n_estimators=100)
regressor.fit(X, y)
```

```
▾ RandomForestRegressor
RandomForestRegres
sor()
```

In [96]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor , X = X_train, y = y_train, cv = 5)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 85.49 %
Standard Deviation: 4.58 %
```

In [97]:

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(criterion = 'friedman_mse',splitter = 'random',max_depth=6,
regressor.fit(X, y)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(criterion='friedman_mse', m
ax_depth=6,
                      min_samples_split=4, splitt
er='random')
```

In [98]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor , X = X_train, y = y_train, cv = 5)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 75.91 %
Standard Deviation: 7.66 %
```

In [99]:

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(criterion = 'friedman_mse',splitter = 'random',max_depth=5,
regressor.fit(X, y)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(criterion='friedman_mse', m
ax_depth=5,
                      min_samples_split=4, splitt
er='random')
```

In [100]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor , X = X_train, y = y_train, cv = 5)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 80.64 %
Standard Deviation: 4.61 %
```

Random Forest Accuracy is more

Accuracy is 85%

In [ ]: