ProblemStatement: Which model will sutable(bestfit) for the given dataset

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Data collection

In [2]:

```
df=pd.read_csv(r"C:\Users\magam\Downloads\insurance.csv")
df.head()
```

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

Data preprocessing

In [3]:

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

Out[3]:

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

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
```

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['bmi'].value_counts()
```

Out[6]:

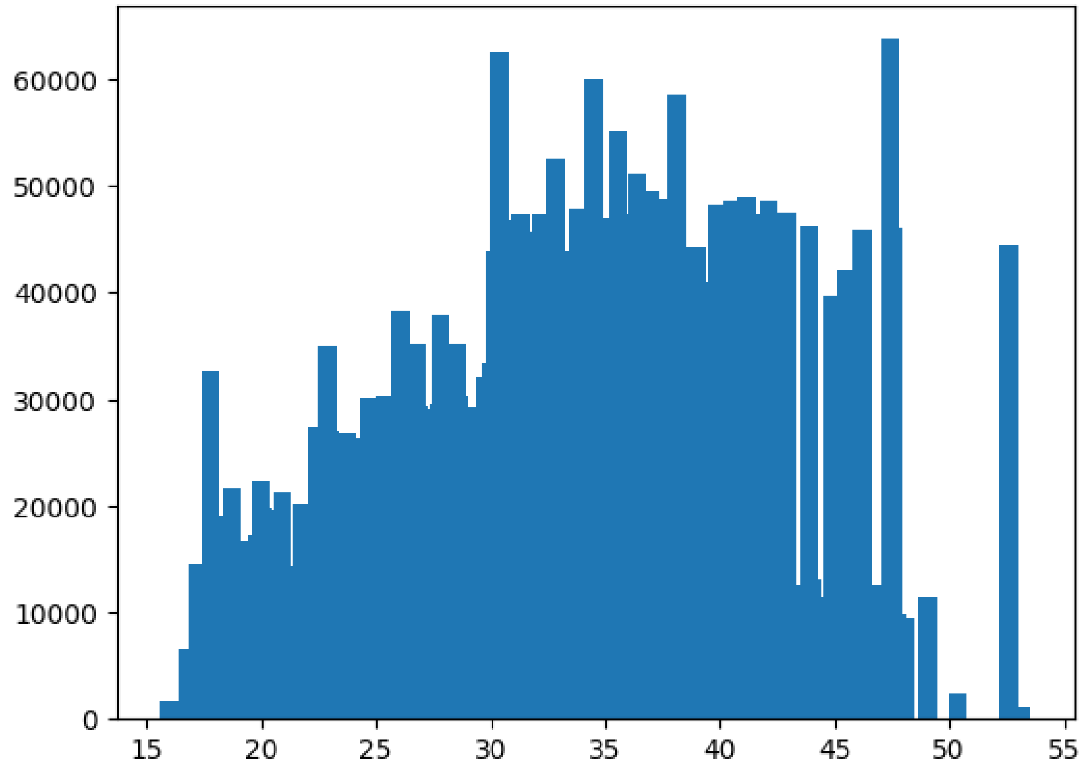
```
bmi
32.300    13
28.310     9
30.495     8
30.875     8
31.350     8
          ..
46.200     1
23.800     1
44.770     1
32.120     1
30.970     1
Name: count, Length: 548, dtype: int64
```

In [62]:

```
x=df['bmi']
y=df['charges']
plt.bar(x,y)
```

Out[62]:

```
<BarContainer object of 1338 artists>
```



In [8]:

```
df.columns
```

Out[8]:

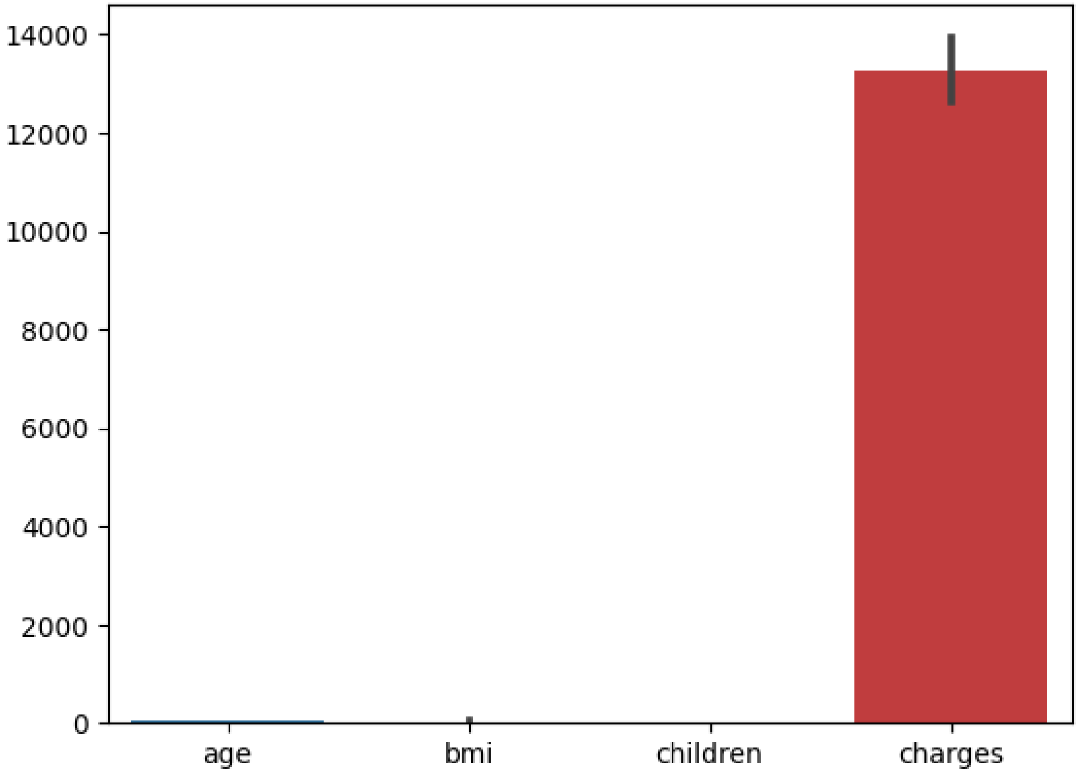
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'],
dtype='object')
```

In [9]:

```
sns.barplot(df)
```

Out[9]:

```
<Axes: >
```



In [10]:

```
df.describe()
```

Out[10]:

	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 [11]:

```
df.columns
```

Out[11]:

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

In [12]:

```
sex={"sex":{"male":1,"female":0}}
df=df.replace(sex)
df
```

Out[12]:

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

1338 rows × 7 columns

In [13]:

```
smoker={"smoker":{"yes":1,"no":0}}
df=df.replace(smoker)
df
```

Out[13]:

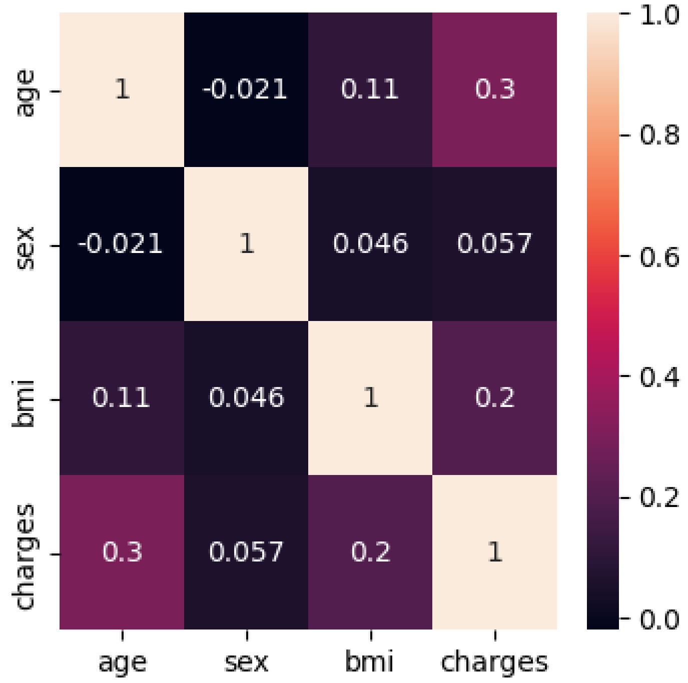
```
      age  sex     bmi  children  smoker     region      charges
   0   19    0  27.900         0       1  southwest  16884.92400
   1   18    1  33.770         1       0  southeast   1725.55230
   2   28    1  33.000         3       0  southeast   4449.46200
   3   33    1  22.705         0       0  northwest  21984.47061
   4   32    1  28.880         0       0  northwest   3866.85520
 ...  ...  ...     ...       ...     ...        ...          ...
1333   50    1  30.970         3       0  northwest  10600.54830
1334   18    0  31.920         0       0  northeast   2205.98080
1335   18    0  36.850         0       0  southeast   1629.83350
1336   21    0  25.800         0       0  southwest   2007.94500
1337   61    0  29.070         0       1  northwest  29141.36030

1338 rows × 7 columns
```

In [14]:

```
idf=df[['age','sex','bmi','charges']]
plt.figure(figsize=(4,4))
sns.heatmap(idf.corr(),annot=True)
```

Out[14]:

```
<Axes: >
```



In [20]:

```
x=df[['age','sex','bmi','children','smoker']]
y=df['charges']
```

LINEAR REGRESSION

In [21]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
lr=LinearRegression()
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100

```

In [22]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.intercept_)
coeff_df=pd.DataFrame(lr.coef_,x.columns,columns=['coefficient'])
coeff_df
```

```
-10719.483493479494
```

Out[22]:

	coefficient
age	259.757578
sex	18.216925
bmi	277.903898
children	461.169867
smoker	23981.741027

In [24]:

```
score=lr.score(x_test,y_test)
print(score)
```

```
0.780095696440481
```

In [25]:

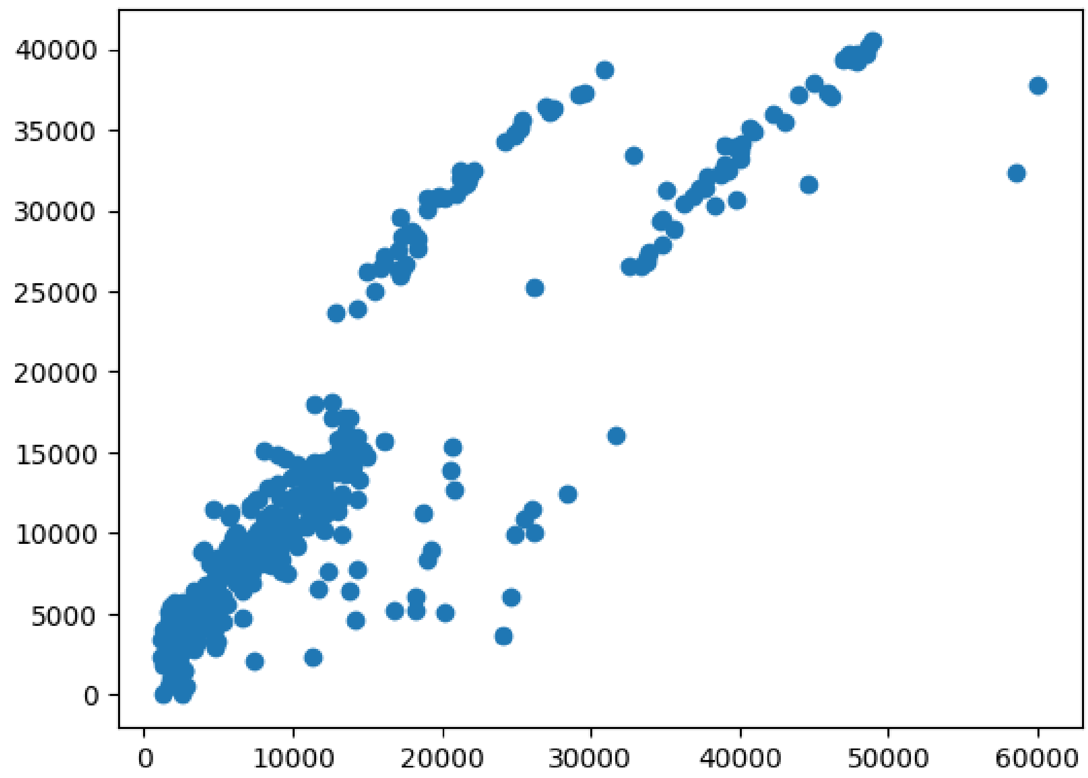
```
predictions=lr.predict(x_test)
```

In [26]:

```
plt.scatter(y_test,predictions)
```

Out[26]:

```
<matplotlib.collections.PathCollection at 0x1a3c13ada50>
```



In [27]:

```
x=np.array(df['smoker']).reshape(-1,1)
y=np.array(df['charges']).reshape(-1,1)
df.dropna(inplace=True)
```

In [28]:

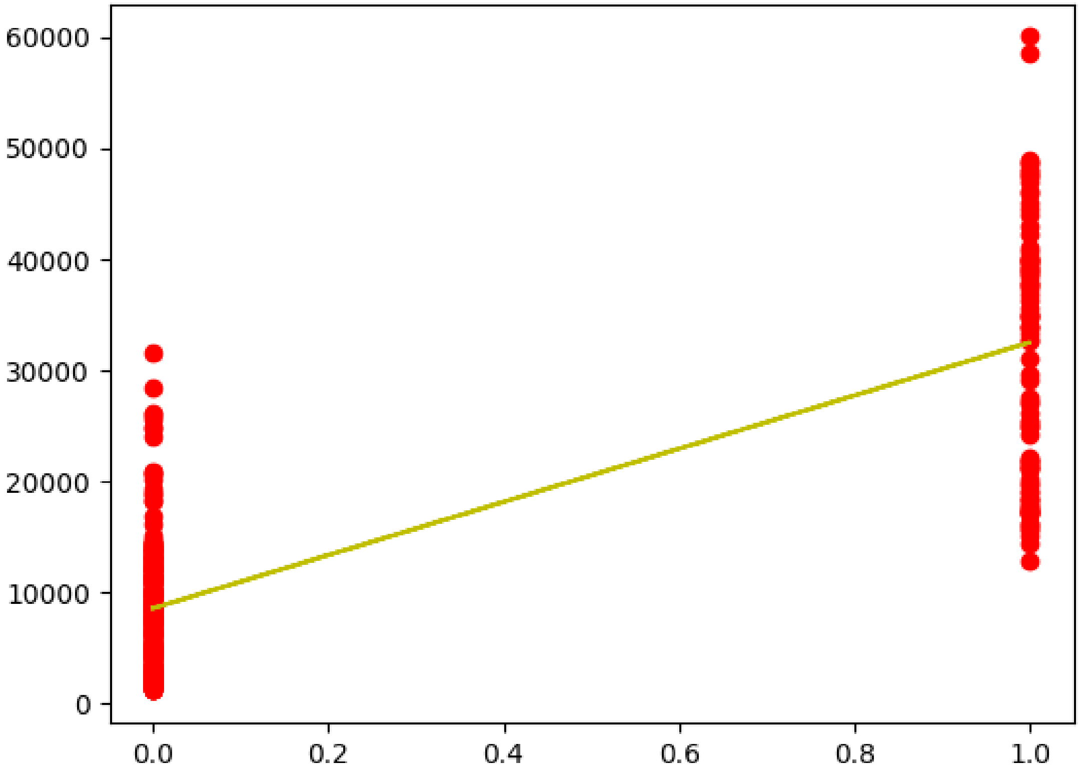
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100
lr.fit(x_train,y_train)
lr.fit(x_train,y_train)
```

Out[28]:

```
▾ LinearRegression
LinearRegression()
```

In [29]:

```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='r')
plt.plot(x_test,y_pred,color='y')
plt.show()
```



LOGISTIC REGRESSION

In [43]:

```
from sklearn.linear_model import LogisticRegression
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100
x=np.array(df['charges']).reshape(-1,1)
y=np.array(df['smoker']).reshape(-1,1)
df.dropna(inplace=True)
lg=LogisticRegression(max_iter=1000)
```

In [44]:

```
lg.fit(x_train,y_train)
```

```
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-package
s\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape
of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[44]:

```
▾       LogisticRegression
LogisticRegression(max_iter=1000)
```

In [45]:

```
score=lg.score(x_test,y_test)
print(score)
```

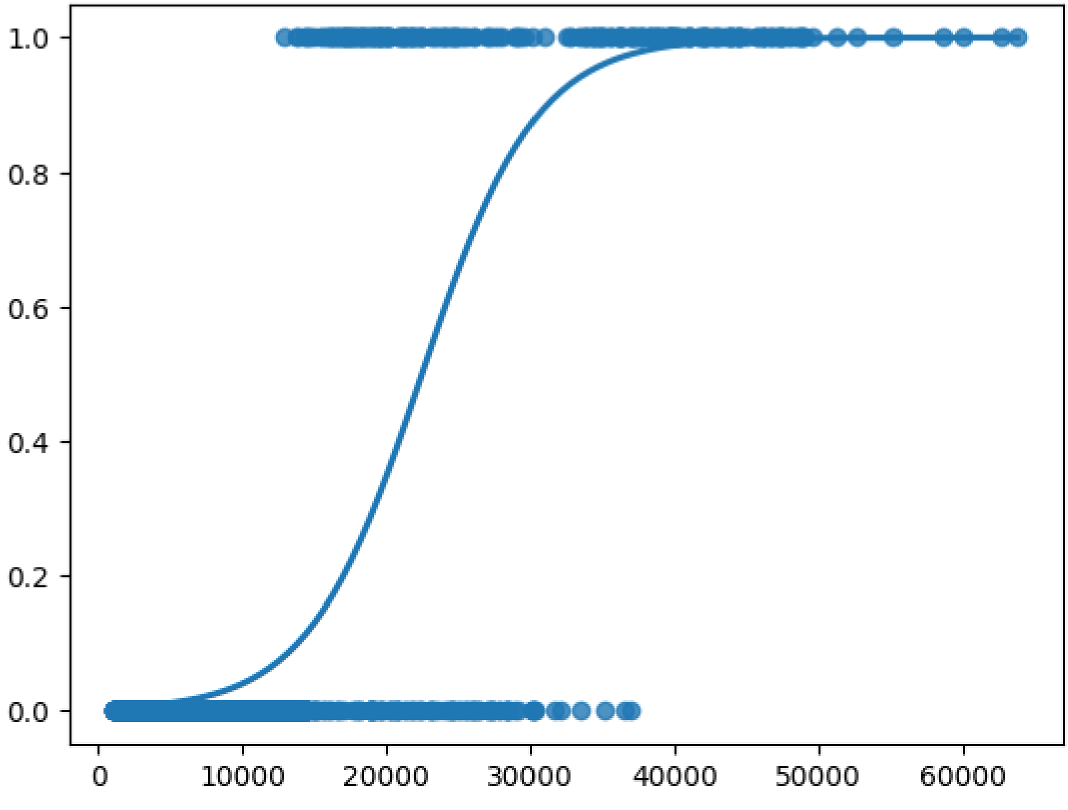
```
0.900497512437811
```

In [46]:

```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[46]:

```
<Axes: >
```



DECISION TREE

In [49]:

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
clf.fit(x_train,y_train)
```

Out[49]:

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [50]:

```
clf.fit(x_train,y_train)
```

Out[50]:

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [51]:

```
score=clf.score(x_test,y_test)
print(score)
```

```
0.8955223880597015
```

RANDOM FOREST

In [52]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
C:\Users\magam\AppData\Local\Temp\ipykernel_920\2210184639.py:3: DataCo
nversionWarning: A column-vector y was passed when a 1d array was expec
ted. Please change the shape of y to (n_samples,), for example using ra
vel().
  rfc.fit(x_train,y_train)
```

Out[52]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [53]:

```
params={'max_depth':[2,3,5,10,20],
        'min_samples_leaf':[5,10,20,50,100,200],
        'n_estimators':[10,25,30,50,100,200]}
```

In [54]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [55]:

```
grid_search.fit(x_train,y_train)
```

```
Please change the shape of y to (n_samples,), for example using ra
vel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\sklearn\model_selection\_validation.py:686: DataConversionW
arning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ra
vel().
  estimator.fit(X_train, y_train, **fit_params)

C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\sklearn\model_selection\_validation.py:686: DataConversionW
arning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ra
vel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\sklearn\model_selection\_validation.py:686: DataConversionW
arning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ra
```

In [56]:

```
grid_search.best_score_
```

Out[56]:

```
0.9337606837606838
```

In [57]:

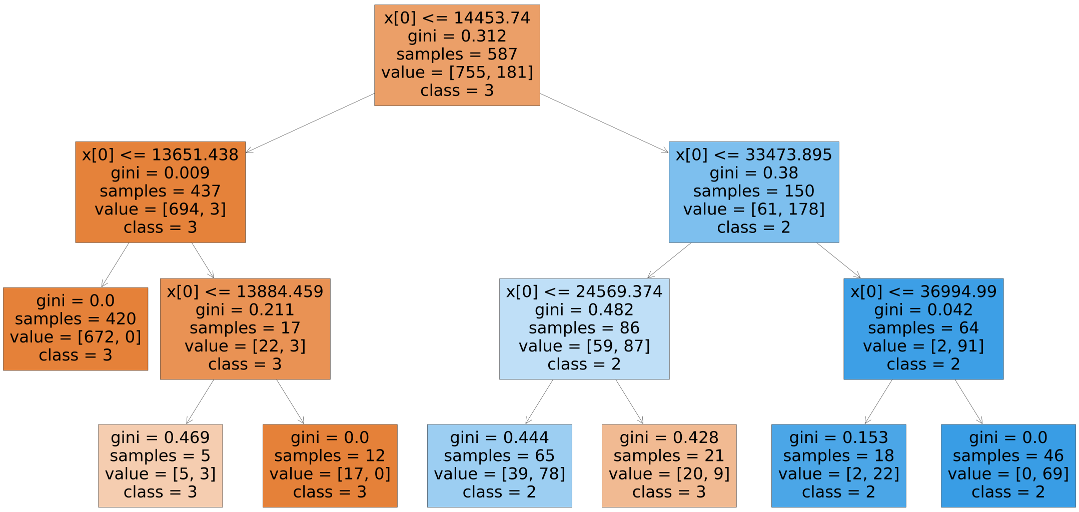
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[57]:

```
▾                         RandomForestClassifier
RandomForestClassifier(max_depth=3, min_samples_leaf=5, n_estimators=1
0)
```

In [58]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['3','2','1','0'],filled=True);
```



In [59]:

```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.8955223880597015
```

CONCLUSION

By analysing the data with Linear,logistic,DecissionTree,RandomForest models , I can got 78% for Linear , 90% for Logistic , 89% for DecissionTree and 89% for Randomforest. by this I conclude that Logistical model is the bestfit model.