Cost of Treatment of Patient Prediction Based on Medical Cost Personal Datasets

Part 1 - DEFINE

---Step1.Define the problem-----> Accurately Predict the insurance costs, based on medical cost personal dataset

```


import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import PolynomialFeatures
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


```

Part 2 - DISCOVER ----Step2.Load Dataset---->Check Head, info and describe , shape of dataset by query

```
df= pd.read_csv('/kaggle/input/insurance/insurance.csv')
```

```
  df.head(10)
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400

```
1 df.describe()
```

	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

```
1 df.info()
```

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

```
1 print('Number of rows and columns in the data set: ',df.shape)
```

```
Number of rows and columns in the data set:  (1338, 7)
```

Now we have imported dataset. When we look at the shape of dataset it has return as (1338,7).So there are m=1338 training exaple and n=7 independent variable. The target variable here is charges and remaining six variables such as age, sex, bmi, children, smoker, region are independent variable.

----Step3.Clean Dataset---

```
1 df.isnull().sum(axis=0)
```

```
age         0
sex         0
```

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

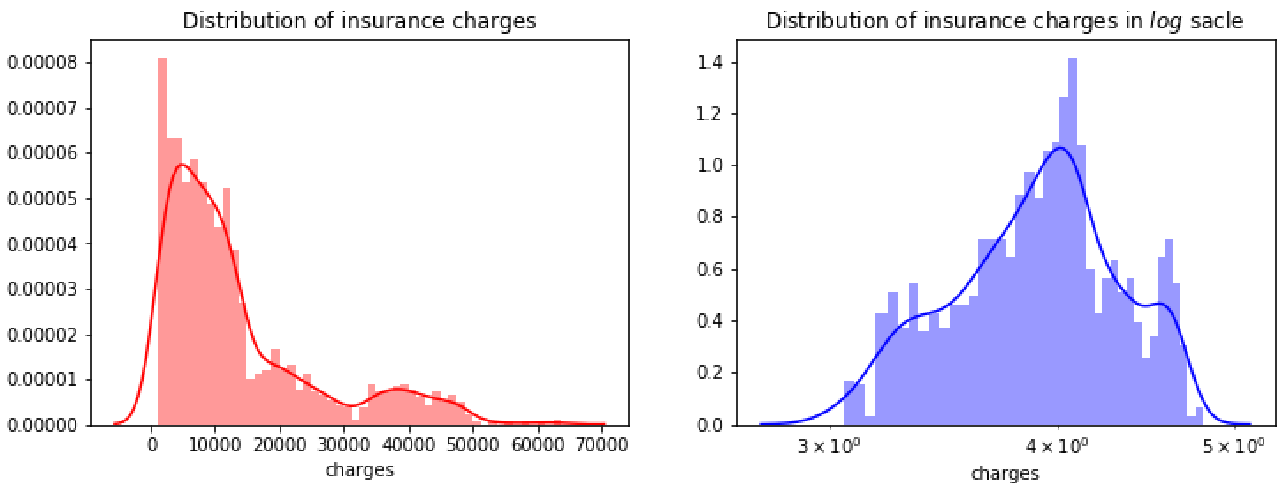
---Step4.Explore the Data (EDA)--

a.Visualizing the Charges data Target Variable by using distplot

```
f= plt.figure(figsize=(12,4))
ax=f.add_subplot(121)
sns.distplot(df['charges'],bins=50,color='r',ax=ax)
ax.set_title('Distribution of insurance charges')

ax=f.add_subplot(122)
sns.distplot(np.log10(df['charges']),bins=40,color='b',ax=ax)
ax.set_title('Distribution of insurance charges in $log$ sacle')
ax.set_xscale('log')
plt.show()

```

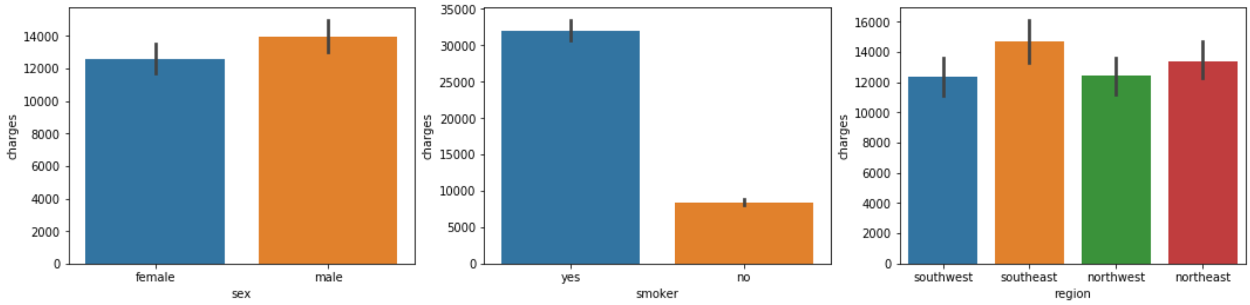


b.Visualizing categorical data by using bar plot

- sex
- smoker
- region

```
plt.figure(figsize=(18,4))
plt.subplot(131)
sns.barplot(x='sex', y='charges', data=df)
plt.subplot(132)
sns.barplot(x='smoker', y='charges', data=df)
plt.subplot(133)
```

```
7 sns.barplot(x='region', y='charges', data=df)
8 plt.show()
```

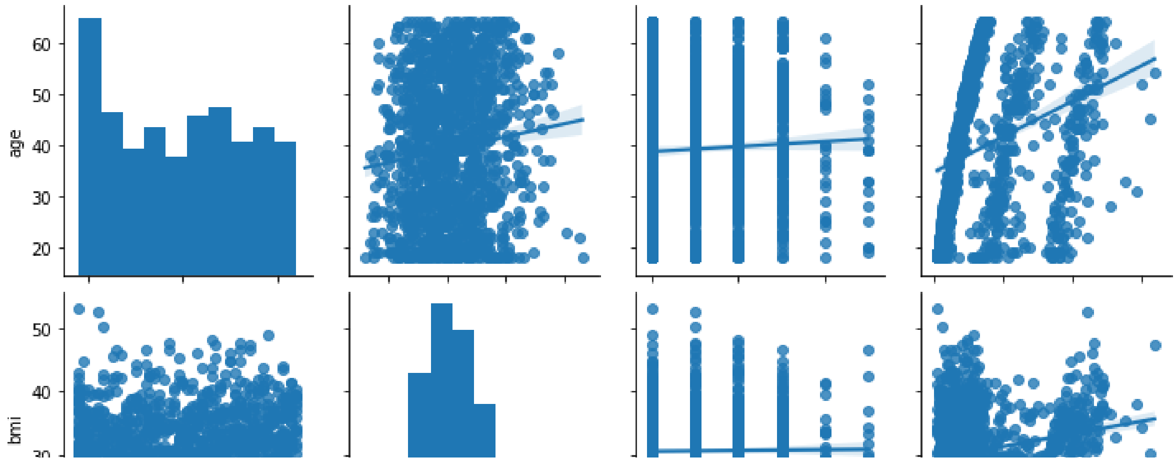


c.Visualizing Numerical data by using pairplot

- age
- bmi
- children
- charges

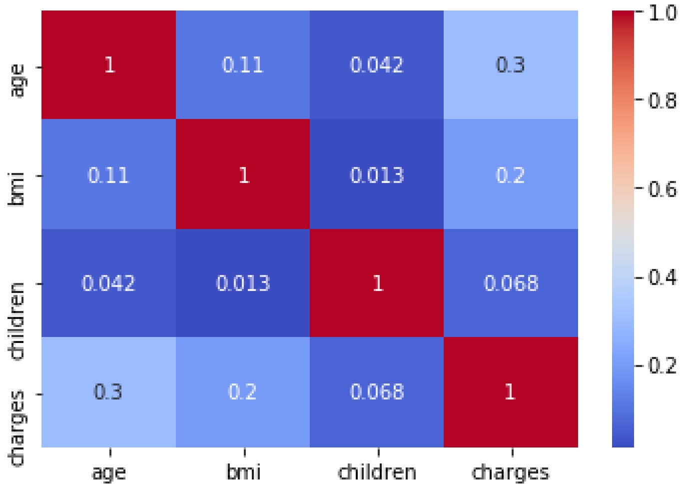
```
1 sns.pairplot(df,kind="reg")
```

```
<seaborn.axisgrid.PairGrid at 0x7fa85595e7b8>
```




```
1 sns.heatmap(df.corr(), cmap='coolwarm',annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa855880358>
```



--Step5.Label Encoding for Catogorical data---

Label encoding refers to transforming the word labels into numerical form so that the algorithms can understand how to operate on them.

```
1 df['sex']=df['sex'].map({'male':1, 'female':0})
2 df['smoker']=df['smoker'].map({'yes':1,'no':0})
```

```
1
2 df = pd.get_dummies(df, columns=['region'], drop_first=True)
3 df.head()
```

	age	sex	bmi	children	smoker	charges	region_northwest	region_southe
1	18	1	33.770	1	0	1725.55230	0	

Part 3 DEVELOP **Train Test split**

```
from sklearn.model_selection import train_test_split
X = df.drop('charges',axis=1)
y = df['charges']

X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=0)
```

```
lr = LinearRegression()
lr.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)
print(lr.score(X_test,y_test))
```

```
0.7958786376014415
```

Now lets add Polynmial Feature and look at the result

```
X = df.drop(['charges','region_northwest','region_southeast','region_southwest'], axis
Y = df.charges



quad = PolynomialFeatures (degree = 2)
x_quad = quad.fit_transform(X)

X_train,X_test,Y_train,Y_test = train_test_split(x_quad,Y, random_state = 0)

plr = LinearRegression().fit(X_train,Y_train)

Y_train_pred = plr.predict(X_train)
Y_test_pred = plr.predict(X_test)

print(plr.score(X_test,Y_test))
```

```
0.8849197344147235
```

No tets try out with Random Forest

```
forest = RandomForestRegressor(n_estimators = 100,
                              criterion = 'mse',
                              random_state = 1,
                              n_jobs = -1)
forest.fit(X_train,y_train)
```

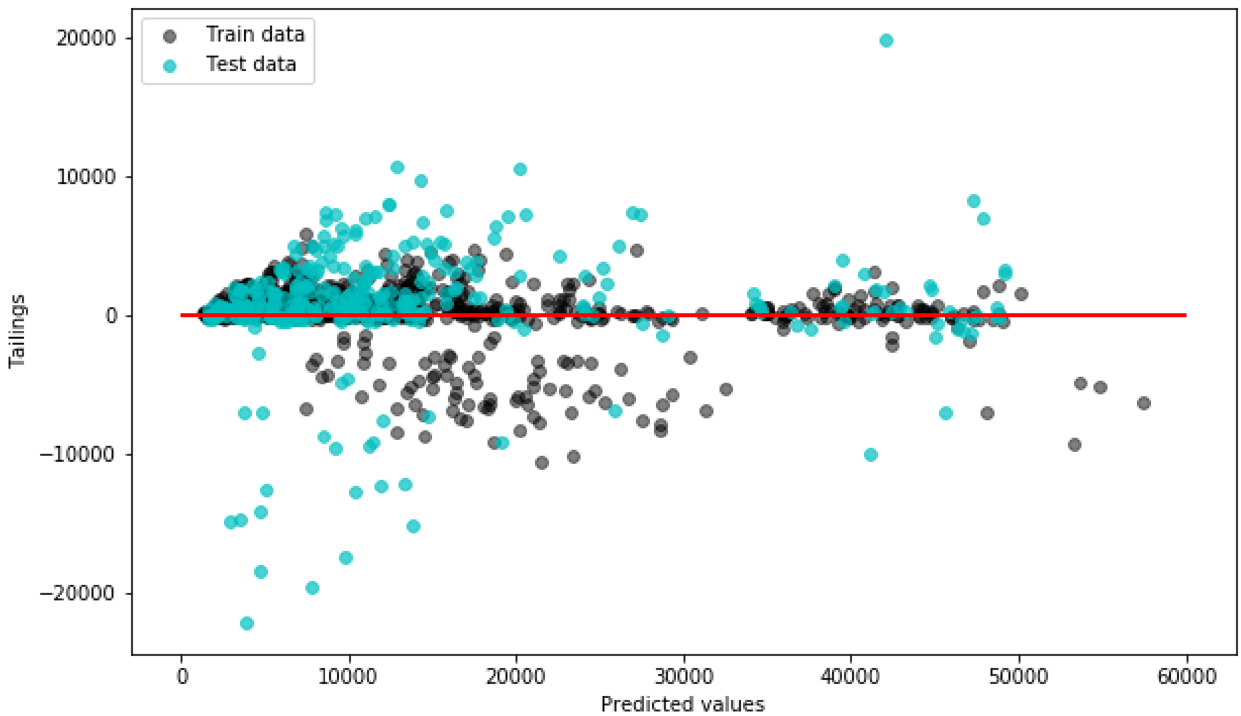
```
forest_train_pred = forest.predict(X_train)
forest_test_pred = forest.predict(X_test)

print('MSE train data: %.3f, MSE test data: %.3f' % (
mean_squared_error(y_train,forest_train_pred),
mean_squared_error(y_test,forest_test_pred)))
print('R2 train data: %.3f, R2 test data: %.3f' % (
r2_score(y_train,forest_train_pred),
r2_score(y_test,forest_test_pred)))
```

```
MSE train data: 3969212.165, MSE test data: 20081745.321
R2 train data: 0.972, R2 test data: 0.872
```

```
plt.figure(figsize=(10,6))

plt.scatter(forest_train_pred,forest_train_pred - y_train,
          c = 'black', marker = 'o', s = 35, alpha = 0.5,
          label = 'Train data')
plt.scatter(forest_test_pred,forest_test_pred - y_test,
          c = 'c', marker = 'o', s = 35, alpha = 0.7,
          label = 'Test data')
plt.xlabel('Predicted values')
plt.ylabel('Tailings')
plt.legend(loc = 'upper left')
plt.hlines(y = 0, xmin = 0, xmax = 60000, lw = 2, color = 'red')
plt.show()
```



*Still there is chances off improvement Hope to You attain 100% accuracy next time * In my opinian you go ahead with other regression algoritham available , with parameter tuning can acheive geat result