Software Development for A.I.

Project Increment - 1

Title: **Health Insurance Premium Prediction using AWS Cloud**

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
```

Important Libraries

1. Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

2. Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series

3. Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

4. scikit-learn

Scikit-learn is an open source data analysis library, and the gold standard for Machine Learning (ML) in the Python ecosystem

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

```
# Reading the Dataset
insurance = pd.read_csv('/content/drive/MyDrive/insurance.csv')
```

```
# Quick glance at the data
insurance
```

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86
...	...	...	...	...	...	...	...
1333	50	male	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83

Performing Exploratory Data Analysis

```
#Checking for null values in the Dataset
sns.heatmap(insurance.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5efc1e4290>
```



```
# Grouping by region to see any relationship between region and charges
region = insurance.groupby(by='region').mean()
region
# It reveals that the south east region has higher BMI values when compared to northeast
```

region	age	bmi	children	expenses
northeast	39.268519	29.176235	1.046296	13406.384691
northwest	39.196923	29.201846	1.147692	12417.575169
southeast	38.939560	33.359341	1.049451	14735.411538

```
# Grouping by age
age = insurance.groupby(by='age').mean()
age
```

34	30.273077	1.153846	11613.528462
35	31.392000	1.680000	11307.183200
36	29.368000	1.240000	12204.477600
37	31.216000	1.520000	18019.911600
38	29.004000	1.480000	8102.732800
39	29.908000	2.200000	11778.243600
40	30.144444	1.592593	11772.251481
41	31.518519	1.407407	9653.745556
42	30.337037	1.000000	13061.038519
43	30.207407	1.629630	19267.279630
44	30.848148	1.222222	15859.397037
45	29.782759	1.482759	14830.199310
46	31.341379	1.620690	14342.591379
47	30.655172	1.379310	17653.999655
48	31.927586	1.310345	14632.500000
49	30.314286	1.500000	12696.006071
50	31.134483	1.310345	15663.003103
51	31.731034	1.103448	15682.255517
52	32.941379	1.482759	18256.270345
53	30.371429	1.250000	16020.930357
54	31.232143	1.428571	18758.546429
55	31.950000	0.961538	16164.545000

```
# Conversion of categorical value to numerical value to avoid training issues with the machine learning model
insurance['sex'] = insurance['sex'].apply(lambda x: 0 if x == 'female' else 1)
insurance['smoker'] = insurance['smoker'].apply(lambda x: 1 if x == 'yes' else 0)


# Check unique values in 'region' column
insurance['region'].unique()
```

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

```
region_dummies = pd.get_dummies(insurance['region'], drop_first = True)


insurance = pd.concat([insurance, region_dummies], axis = 1)


insurance.describe()
```

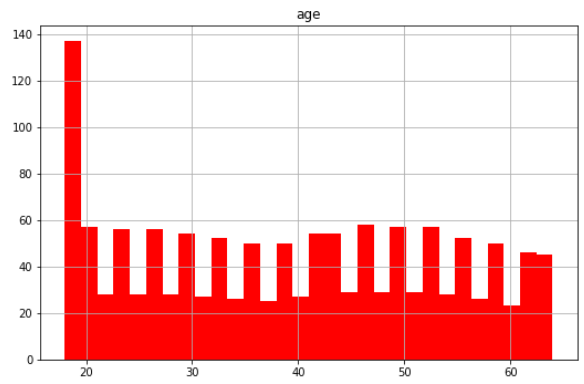
	age	sex	bmi	children	smoker	expenses
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.665471	1.094918	0.204783	13270.422414
std	14.049960	0.500160	6.098382	1.205493	0.403694	12110.011240
min	18.000000	0.000000	16.000000	0.000000	0.000000	1121.870000
25%	27.000000	0.000000	26.300000	0.000000	0.000000	4740.287500
50%	39.000000	1.000000	30.400000	1.000000	0.000000	9382.030000

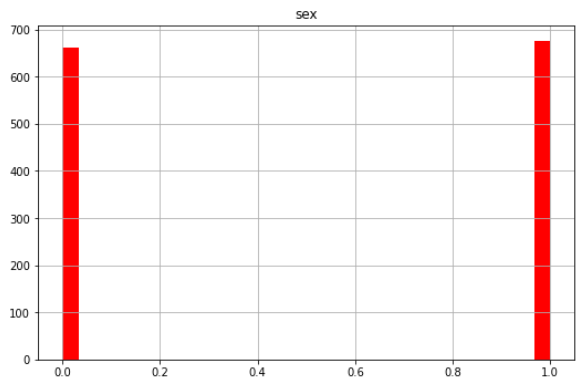
Plotting the Dataset graph using Seaborn

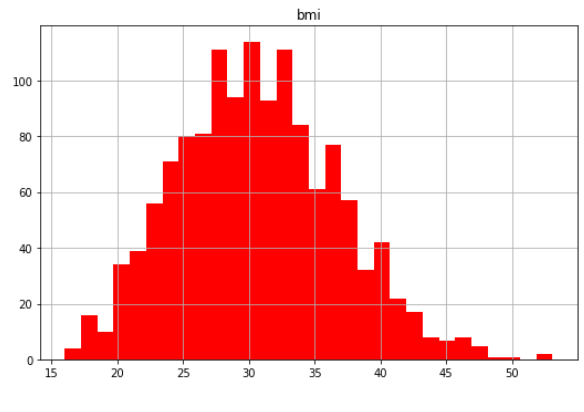
1. Graphs showing the overview of aspects like Age, Sex, BMI, Children, Smoker, Expenses

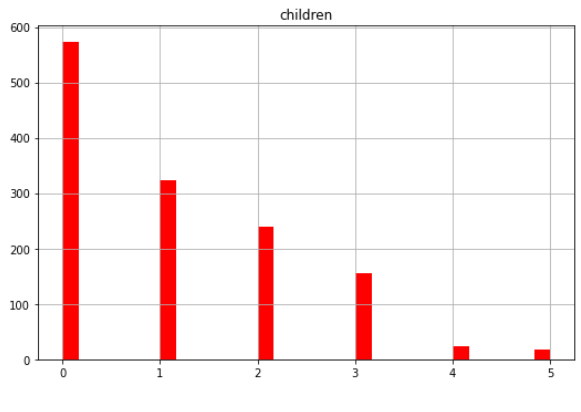
```
insurance[['age', 'sex', 'bmi', 'children', 'smoker', 'expenses']].hist(bins = 30, figsize = (20,20), color = 'r')
```

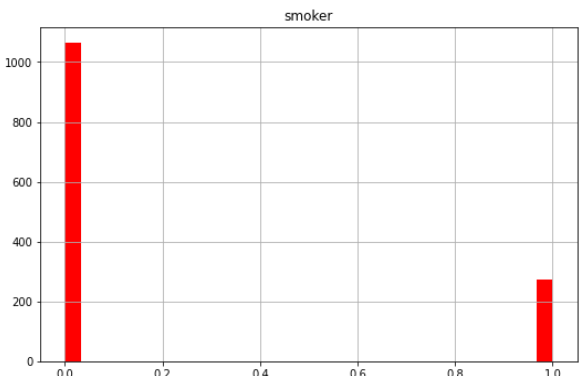
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f5ef94cad10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f5ef9485450>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f5ef943ca50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f5ef93e8b50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f5ef93b8690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f5ef936ec90>]],
      dtype=object)
```

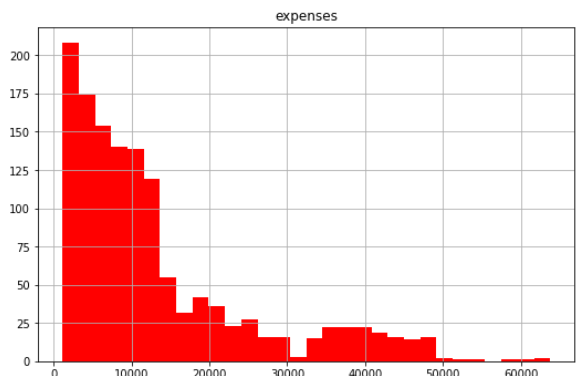






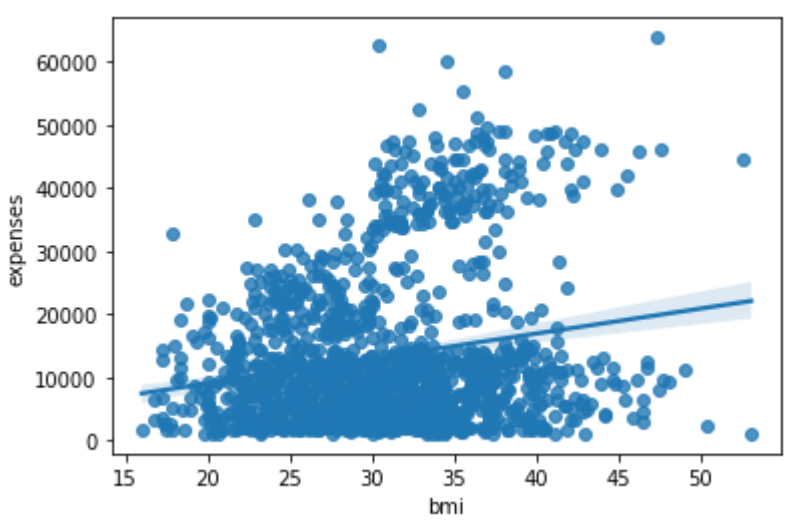







```
# Plotting age and expenses
sns.regplot(x = 'age', y = 'expenses', data = insurance)
plt.show()
```


```
# Plotting BMI and expenses
sns.regplot(x = 'bmi', y = 'expenses', data = insurance)
plt.show()
```




```
# Finding the corealation among them
corelation = insurance.corr()
corelation
```

	age	sex	bmi	children	smoker	expenses	northwest	southeast	southwest
age	1.000000	-0.020856	0.109341	0.042469	-0.025019	0.299008	-0.000407	-0.011642	0.010016
sex	-0.020856	1.000000	0.046380	0.017163	0.076185	0.057292	-0.011156	0.017117	-0.004184
bmi	0.109341	0.046380	1.000000	0.012645	0.003968	0.198576	-0.135992	0.270144	-0.006398
children	0.042469	0.017163	0.012645	1.000000	0.007673	0.067998	0.024806	-0.023066	0.021914
smoker	-0.025019	0.076185	0.003968	0.007673	1.000000	0.787251	-0.036945	0.068498	-0.036945
expenses	0.299008	0.057292	0.198576	0.067998	0.787251	1.000000	-0.039905	0.073982	-0.043210
northwest	-0.000407	-0.011156	-0.135992	0.024806	-0.036945	-0.039905	1.000000	-0.346265	-0.320829
southeast	-0.011642	0.017117	0.270144	-0.023066	0.068498	0.073982	-0.346265	1.000000	-0.346265
southwest	0.010016	-0.004184	-0.006398	0.021914	-0.036945	-0.043210	-0.320829	-0.346265	1.000000

```
plt.figure(figsize = (10,10))
sns.heatmap(corelation , annot = True)
```