Medical_Cost_Analysis.ipynb

Reading file using pandas

```
import pandas as pd
data =pd.read_csv ("/content/100636")
data.head()
```

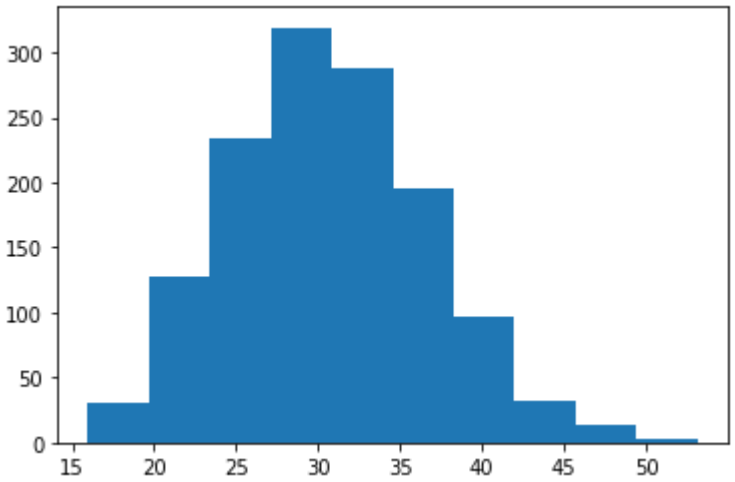
	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

importing required Libraries

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

Distribution of bmi aka body mass index

```
plt.hist (data['bmi'])
plt.show ()
plt.figure(figsize = (7, 6))
sns.countplot(data['bmi'])
plt.title('bmi Value Counts')
```

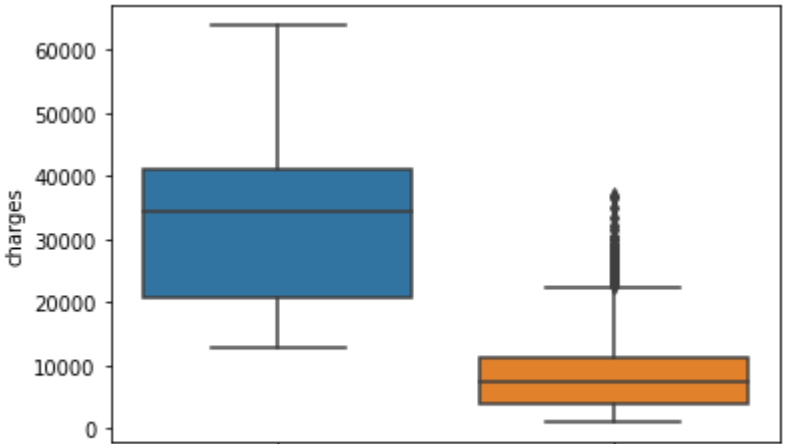



```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
  FutureWarning
Text(0.5, 1.0, 'bmi Value Counts')
```

Relationship between 'smokers 'and 'charges'

```
sns.boxplot (x=data['smoker'],y=data['charges'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feed6a78150>
```



Relationship between Smoker and Region

```
%matplotlib inline

sns.histplot(binwidth=0.5, x="region", hue="smoker", data=data, stat="count", multiple="st
```

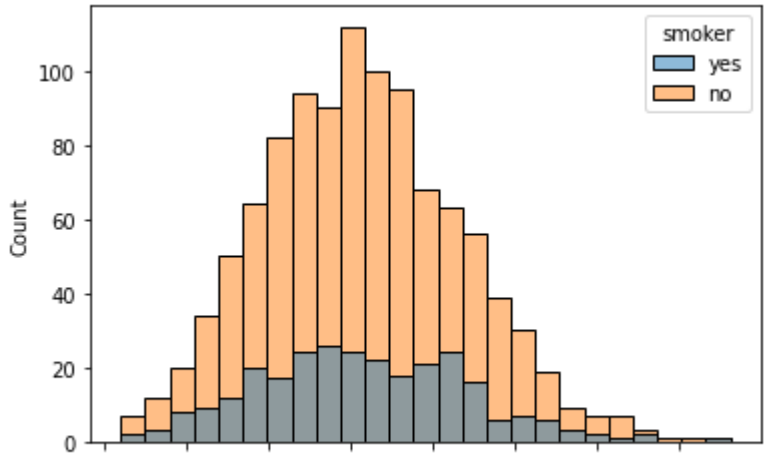
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feed64d2610>
```

Relationship between BMI and SEX

```
sns.histplot(x=data['bmi'],hue=data['smoker'])

#plt.figure(figsize = (7, 6))
#sns.set_style('whitegrid')
#sns.catplot( 'sex',col = 'bmi', hue = None, data = data, kind = 'count')
```

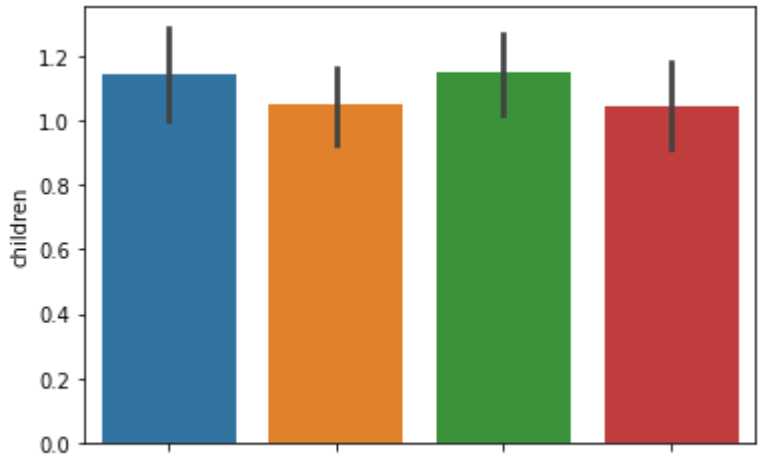
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feed6491110>
```



Region with children most

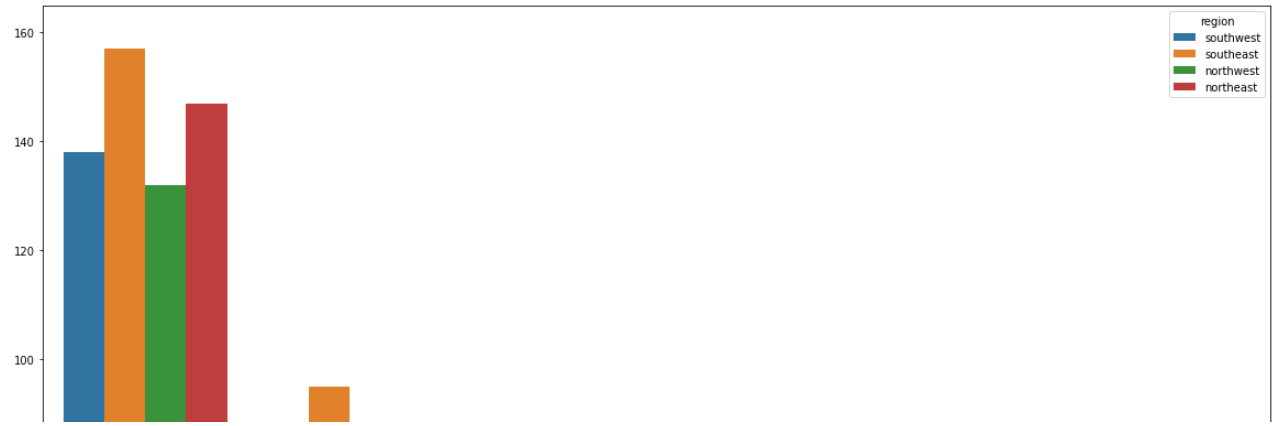
```
sns.barplot( y="children", x="region", data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feed64308d0>
```




```
plt.figure(figsize = (20, 15))
sns.countplot(data['children'],hue=data['region'])
```

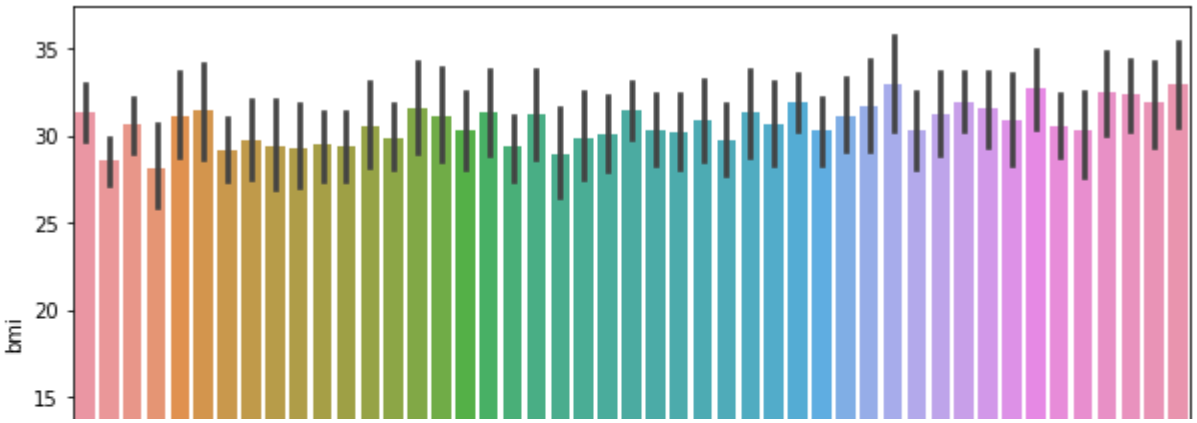
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feed62e8790>
```



Relationship between BMI and AGE

```
plt.figure(figsize = (10, 6))
sns.barplot( x=data['age'],y =data['bmi'])
#plt.hist( x=data["age"],y =data["bmi"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feed625dd50>
```



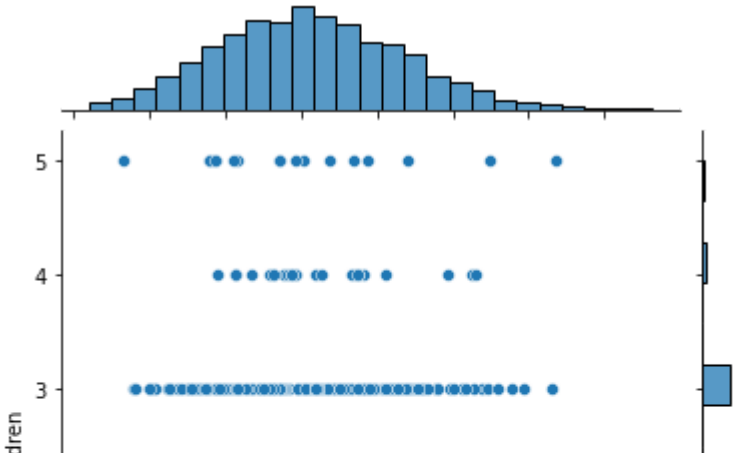
Relationship between bmi, children

```
data.loc[(data['age'] >= 18) &(data['age'] <= 18)  ,'age_category'] = 'children'
data.loc[(data['age'] >= 25) & (data['age'] <= 40), 'age_category'] = 'adults'
data.loc[data['age'] > 50, 'age_category'] = 'seniors'
data['age_category'].value_counts()
#sns.barplot( x=data['age_category']['youth'] ,y =data['bmi'])

plt.figure(figsize = (5, 10))

sns.jointplot ( x=data['bmi'], y=data['children'])
```

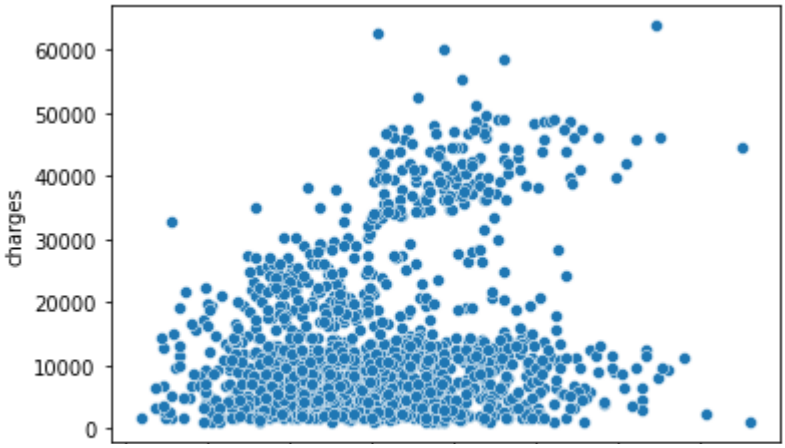
```
<seaborn.axisgrid.JointGrid at 0x7feed6027550>
<Figure size 360x720 with 0 Axes>
```



to detect outliers in bmi

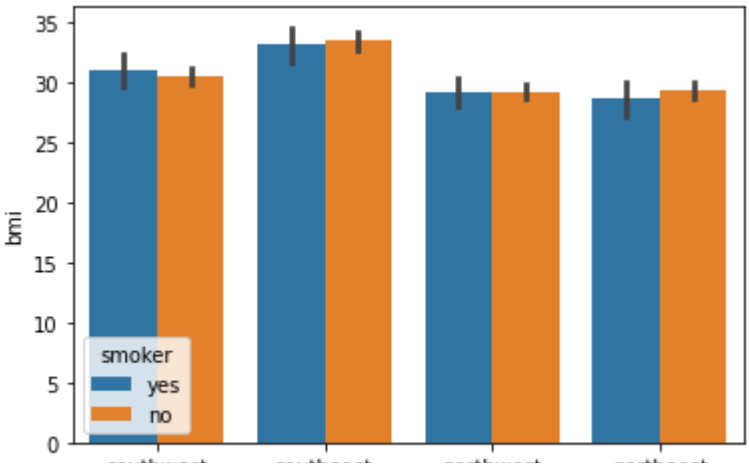
```
#some outlier are in bmi
sns.scatterplot( x=data["bmi"],y =data["charges"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feed5f0f050>
```




```
sns.barplot(x = data['region'],y = data['bmi'],hue = data['smoker'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feed5e96390>
```



Data Processing

Using Label Encoding and One-Hot Encoding techniques to deal with categorical variables.

● Spliting your dataset into X_train,X_test, y_train, y_test. ● Scaleing the dataset by normalizing it(Min-Max Scaling or Standard Scaling).

```
data.info()
from sklearn.preprocessing import LabelEncoder ,minmax_scale
from sklearn.model_selection import train_test_split
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 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
 7   age_category  848 non-null    object 
dtypes: float64(2), int64(2), object(4)
memory usage: 83.8+ KB
```

```
#spliting data for training
X=data.drop('charges',axis =1)
y = data['charges']
encoder = LabelEncoder ()
for i in X.columns :
   X[i]= encoder.fit_transform (X[i])
y =encoder.fit_transform (y)
# Normalize the features dataset and assign it to a variable
X_scaled = minmax_scale(X)
# Create a DataFrame using the new variable
X = pd.DataFrame(X_scaled)
#spliting data
X_train ,X_test ,y_train ,y_test =train_test_split (X,y,train_size = 0.75,random_state =42
```

creating regression models and examining those with cross validation

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression,RidgeClassifier
from sklearn.linear_model   import LinearRegression
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score
import numpy as np
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor


#randomforest_model1 =RandomForestClassifier
randomforest_model =RandomForestRegressor()
svm_model = SVR()
linear_model = LinearRegression ()
mlp_model = MLPRegressor ()
kn_model = KNeighborsRegressor ()
descision_tree_model = DecisionTreeRegressor()
Kfold_validation = KFold(10)
randomforest_results =cross_val_score(randomforest_model,X,y,cv =Kfold_validation )
svm_results =cross_val_score(svm_model,X,y,cv =Kfold_validation)
linear_results =cross_val_score(linear_model,X,y,cv =Kfold_validation )
mlp_results =cross_val_score(mlp_model,X,y,cv = Kfold_validation )
kn_results =cross_val_score(kn_model,X,y,cv =Kfold_validation )
decisiontree_results =cross_val_score(descision_tree_model ,X,y,cv =Kfold_validation )


print(randomforest_results)
print(np.mean(randomforest_results))
print()
print(svm_results)
print(np.mean(svm_results))
print()
print(linear_results)
print(np.mean(linear_results))
print()
print(mlp_results)
print(np.mean(mlp_results))
print()
print(kn_results)
print(np.mean(kn_results))
print()
print(decisiontree_results)
print(np.mean(decisiontree_results))
```

```
[0.8509929  0.8162285  0.65670932 0.70186236 0.8119702  0.87064801
 0.82539153 0.73251693 0.71827913 0.83368691]
0.7818285798471721

[0.20738637 0.20868095 0.16550091 0.19746066 0.21087593 0.20460192
 0.21342078 0.18842339 0.18485953 0.21416289]
0.19953733488595776

[0.8142814  0.7896074  0.64956884 0.68829662 0.80968983 0.85587585
 0.83767942 0.6895444  0.64050622 0.78124154]
0.7556291529017238

[0.30127042 0.29297859 0.19738069 0.26135392 0.23114482 0.29685498
 0.20566389 0.34401118 0.21311937 0.361083  ]
0.27048608510691635
```

```
[0.87003156 0.80506428 0.61721776 0.67497771 0.80274576 0.88828412
 0.82038971 0.70301794 0.68075327 0.78196297]
0.7644445078612694

[0.77593521 0.62484703 0.52001649 0.5621316  0.73675846 0.70495851
 0.57750506 0.60330646 0.61677942 0.6540908 ]
0.6376329037549293
```

```
from sklearn.model_selection import StratifiedKFold
skfold =StratifiedKFold(n_splits=2)

randomforest_score =cross_val_score(randomforest_model,X,y,cv = skfold )
svm_score=cross_val_score(svm_model,X,y,cv = skfold)
linear_score =cross_val_score(linear_model,X,y,cv = skfold )
mlp_score =cross_val_score(mlp_model,X,y,cv = skfold )
kn_score =cross_val_score(kn_model,X,y,cv = skfold )
decisiontree_score =cross_val_score(descision_tree_model,X,y,cv = skfold )


print(np.mean(randomforest_score))
print(np.mean(svm_score))
print(np.mean(linear_score))
print(np.mean(mlp_score))
print(np.mean(kn_score))
print(np.mean(decisiontree_score))
```

```
0.7815499939387178
0.11767915195091744
0.7574675121855717
-0.6231517642295986
0.7426012715615906
0.6517441558393549
```

```
from sklearn.metrics import *
svm_model = SVR()
randomforest_model.fit(X_train,y_train)

svm_model.fit(X_train,y_train)

descision_tree_model .fit(X_train,y_train)

mlp_model.fit(X_train,y_train)

kn_model.fit(X_train,y_train)
linear_model.fit(X_train ,y_train )


svm_pred = svm_model.predict(X_test )
linear_pred = linear_model.predict(X_test)

randomforest_pred = randomforest_model.predict(X_test)

decisiontree_pred = descision_tree_model.predict(X_test)

kn_pred = kn_model.predict(X_test)

mlp_pred = mlp_model.predict(X_test)
```

Model Evaluation ● Evaluating the optimized model using regression model evaluation metrics. (Ex. Mean Squared Error, Mean Absolute Error etc.)

```
svm_report1 = mean_squared_error (y_test ,svm_pred )
svm_report2 = mean_absolute_error (y_test ,svm_pred )
svm_report3 = r2_score (y_test ,svm_pred )
linear_report1 = mean_squared_error (y_test ,linear_pred )
linear_report2 = mean_absolute_error (y_test ,linear_pred )
linear_report3 = r2_score (y_test ,linear_pred )
decisiontree_report1 = mean_squared_error (y_test ,decisiontree_pred  )
decisiontree_report2 = mean_absolute_error (y_test ,decisiontree_pred )
decisiontree_report3 = r2_score (y_test ,decisiontree_pred)
'''naive_report1 = mean_squared_error (y_test ,naive_pred )
naive_report2 = mean_absolute_error (y_test ,naive_pred )
naive_report3 = r2_score (y_test ,naive_pred )'''

mlp_report1 = mean_squared_error (y_test ,mlp_pred )
mlp_report2 = mean_absolute_error (y_test ,mlp_pred )
mlp_report3 = r2_score (y_test ,mlp_pred )
kn_report1 = mean_squared_error (y_test ,kn_pred )
kn_report2 = mean_absolute_error (y_test ,kn_pred )
kn_report3 = r2_score (y_test ,kn_pred )
RFR_report1 = mean_squared_error (y_test ,randomforest_pred )
RFR_report2 = mean_absolute_error (y_test ,randomforest_pred  )
RFR_report3 = r2_score (y_test ,randomforest_pred )
print ("RandomForest_Scores")
print (RFR_report1, RFR_report2,RFR_report3)
print ("KNN_SCORES")
print (kn_report1, kn_report2,r2_score)
print("LmSVM_SCORES:")
print (svm_report1, svm_report2, svm_report3)
print("LINEAR_SCORES:")
print (linear_report1, linear_report2, linear_report3)
print("DECISIONTREE_SCORES:")
print (decisiontree_report1, decisiontree_report2, decisiontree_report3)

#print (, naive_report2, naive_report3)
print("NEURAL_NETWORK_SCORES:")
print (mlp_report1, mlp_report2, mlp_report3)
```

```
RandomForest_Scores
31109.40008172471 86.48214925373135 0.7967112351656114
KNN_SCORES
38261.750567164185 116.60119402985075 <function r2_score at 0x7feed7f54170>
LmSVM_SCORES:
126808.62925922794 307.69397994851033 0.17135111751659438
LINEAR_SCORES:
36555.363478866675 129.0308650012329 0.7611238188403237
DECISIONTREE_SCORES:
45202.18507462687 88.08955223880596 0.7046199429273313
NEURAL_NETWORK_SCORES:
120550.8976389144 289.0190944143418 0.21224314784880094
```

Colab paid products - Cancel contracts here