```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as pt
import warnings
warnings.filterwarnings("ignore")
```

```
df=pd.read_csv("/content/2.medical_insurance.csv")
df
```

	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

```
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
```

```
df.describe()
```

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

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

```
features = ['sex', 'smoker', 'region']

plt.subplots(figsize=(20, 10))
for i, col in enumerate(features):
    plt.subplot(1, 3, i + 1)

    x = df[col].value_counts()
    plt.pie(x.values,
            labels=x.index,
            autopct='%1.1f%%')

plt.show()
```

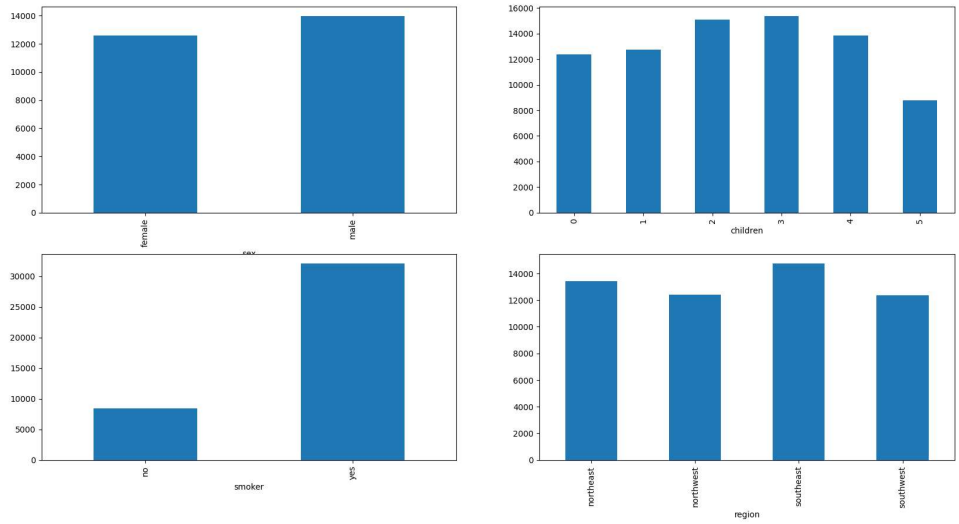
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-27646ee37e92> in <cell line: 3>()
      1 features = ['sex', 'smoker', 'region']
      2
----> 3 plt.subplots(figsize=(20, 10))
      4 for i, col in enumerate(features):
      5         plt.subplot(1, 3, i + 1)

NameError: name 'plt' is not defined
```

SEARCH STACK OVERFLOW

```
import matplotlib.pyplot as plt

features = ['sex', 'smoker', 'region']

plt.subplots(figsize=(20, 10))
for i, col in enumerate(features):
    plt.subplot(1, 3, i + 1)

    x = df[col].value_counts()
    plt.pie(x.values,
            labels=x.index,
            autopct='%1.1f%%')

plt.show()
```

male

```
features = ['sex', 'children', 'smoker', 'region']

plt.subplots(figsize=(20, 10))
for i, col in enumerate(features):
    plt.subplot(2, 2, i + 1)
    df.groupby(col).mean()['charges'].plot.bar()
plt.show()
```

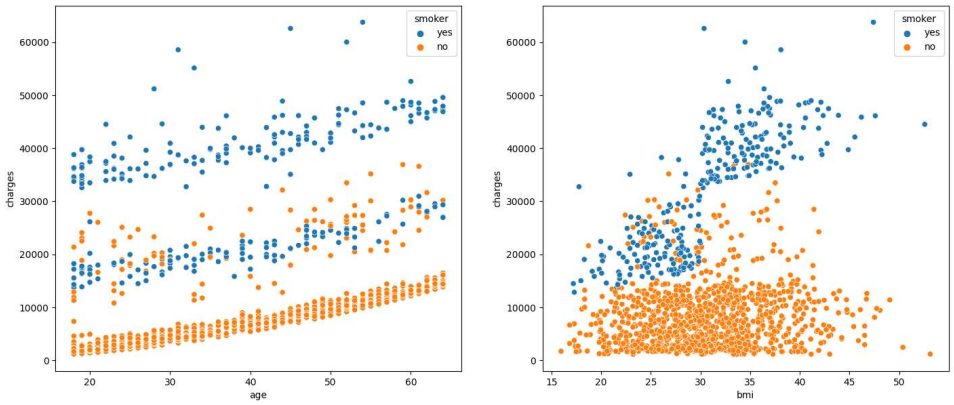



```
features = ['age', 'bmi']

plt.subplots(figsize=(17, 7))
for i, col in enumerate(features):
    plt.subplot(1, 2, i + 1)
    sb.scatterplot(data=df, x=col,
                   y='charges',
                   hue='smoker')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-14-e83d4117eddd> in <cell line: 4>()
      4 for i, col in enumerate(features):
      5         plt.subplot(1, 2, i + 1)
----> 6         sb.scatterplot(data=df, x=col,
      7                                 y='charges',
      8                                 hue='smoker')

NameError: name 'sb' is not defined
```

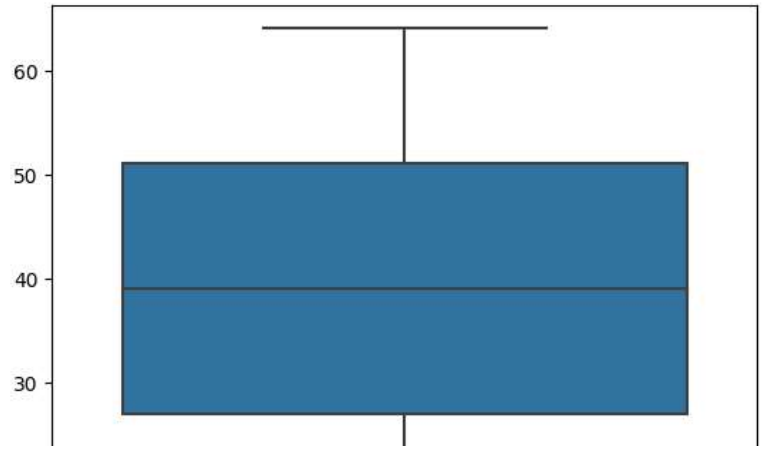
SEARCH STACK OVERFLOW


```
import seaborn as sb
import matplotlib.pyplot as plt

features = ['age', 'bmi']

plt.subplots(figsize=(17, 7))
for i, col in enumerate(features):
    plt.subplot(1, 2, i + 1)
    sb.scatterplot(data=df, x=col, y='charges', hue='smoker')
plt.show()
```



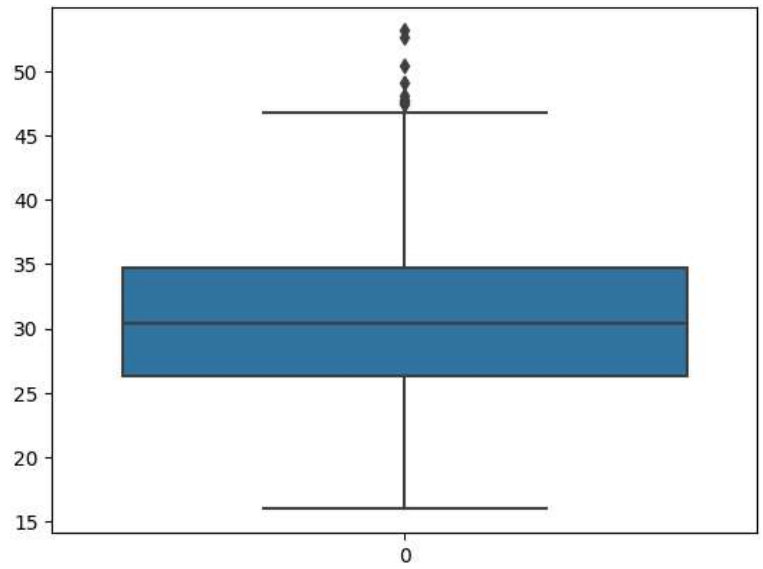

```
df.drop_duplicates(inplace=True)
sns.boxplot(df['age'])
```

```
<Axes: >
```




```
sns.boxplot(df['bmi'])
```

```
<Axes: >
```




```
Q1=df['bmi'].quantile(0.25)
Q2=df['bmi'].quantile(0.5)
Q3=df['bmi'].quantile(0.75)
iqr=Q3-Q1
lowlim=Q1-1.5*iqr
upplim=Q3+1.5*iqr
print(lowlim)
print(upplim)
```

```
13.674999999999994
47.31500000000001
```

```
from feature_engine.outliers import ArbitraryOutlierCapper
arb=ArbitraryOutlierCapper(min_capping_dict={'bmi':13.6749},max_capping_dict={'bmi':47.315})
df[['bmi']]=arb.fit_transform(df[['bmi']])
sns.boxplot(df['bmi'])
```

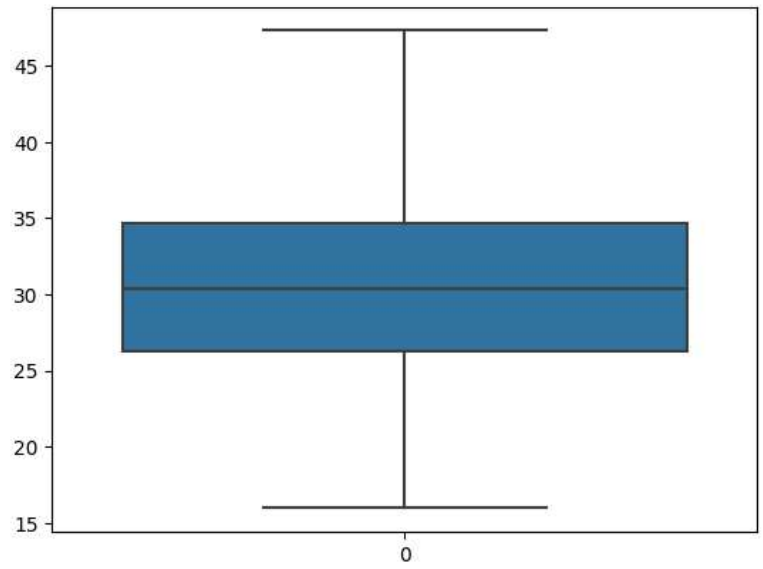
```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-19-2493b5973732> in <cell line: 1>()
----> 1 from feature_engine.outliers import ArbitraryOutlierCapper
      2 arb=ArbitraryOutlierCapper(min_capping_dict={'bmi':13.6749},max_capping_dict=
{'bmi':47.315})
      3 df[['bmi']]=arb.fit_transform(df[['bmi']])
      4 sns.boxplot(df['bmi'])

ModuleNotFoundError: No module named 'feature_engine'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
```

```
!pip install feature-engine
```

```
Collecting feature-engine
  Downloading feature_engine-1.6.2-py2.py3-none-any.whl (328 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 328.9/328.9 kB 3.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.18.2 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.23.5)
Requirement already satisfied: pandas>=1.0.3 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.2.2)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.11.3)
Requirement already satisfied: statsmodels>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (0.14.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.3->feature-e
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.3->feature-engine) (20
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->feature-engi
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->featu
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.1->feature-engin
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.1->feature-en
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.11.1->feature-
Installing collected packages: feature-engine
Successfully installed feature-engine-1.6.2
```

```
from feature_engine.outliers import ArbitraryOutlierCapper
arb=ArbitraryOutlierCapper(min_capping_dict={'bmi':13.6749},max_capping_dict={'bmi':47.315})
df[['bmi']]=arb.fit_transform(df[['bmi']])
sns.boxplot(df['bmi'])
```

```
<Axes: >
```




```
df['bmi'].skew()
df['age'].skew()
```

```
0.054780773126998195
```

```
df['sex']=df['sex'].map({'male':0,'female':1})
df['smoker']=df['smoker'].map({'yes':1,'no':0})
df['region']=df['region'].map({'northwest':0, 'northeast':1,'southeast':2,'southwest':3})
```

```
X=df.drop(['charges'],axis=1)
Y=df[['charges']]
from sklearn.linear_model import LinearRegression,Lasso
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
l1=[]
l2=[]
l3=[]
cvs=0
for i in range(40,50):
  xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.2,random_state=i)
  lrmodel=LinearRegression()
  lrmodel.fit(xtrain,ytrain)
  l1.append(lrmodel.score(xtrain,ytrain))
  l2.append(lrmodel.score(xtest,ytest))
  cvs=(cross_val_score(lrmodel,X,Y,cv=5,)).mean()
  l3.append(cvs)
  df1=pd.DataFrame({'train acc':l1,'test acc':l2,'cvs':l3})
df1
```

	train acc	test acc	cvs
0	0.741659	0.778409	0.74707
1	0.756401	0.706267	0.74707
2	0.729542	0.806239	0.74707
3	0.754260	0.732791	0.74707
4	0.742966	0.779591	0.74707
5	0.753281	0.731769	0.74707
6	0.741261	0.776456	0.74707
7	0.731940	0.796173	0.74707
8	0.751915	0.741742	0.74707
9	0.756348	0.722565	0.74707

```
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.2,random_state=42)
lrmodel=LinearRegression()
lrmodel.fit(xtrain,ytrain)
print(lrmodel.score(xtrain,ytrain))
print(lrmodel.score(xtest,ytest))
print(cross_val_score(lrmodel,X,Y,cv=5,).mean())
```

```
0.7295415541376445
0.806239111557059
0.7470697972809902
```

```
from sklearn.metrics import r2_score
svrmodel=SVR()
svrmodel.fit(xtrain,ytrain)
ypredtrain1=svrmodel.predict(xtrain)
ypredtest1=svrmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain1))
print(r2_score(ytest,ypredtest1))
print(cross_val_score(svrmodel,X,Y,cv=5,).mean())
```

```
-0.10151474302536445
-0.1344454720199666
-0.10374591327267262
```

```
rfmodel=RandomForestRegressor(random_state=42)
rfmodel.fit(xtrain,ytrain)
ypredtrain2=rfmodel.predict(xtrain)
ypredtest2=rfmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain2))
print(r2_score(ytest,ypredtest2))
print(cross_val_score(rfmodel,X,Y,cv=5,).mean())
from sklearn.model_selection import GridSearchCV
estimator=RandomForestRegressor(random_state=42)
param_grid={'n_estimators':[10,40,50,98,100,120,150]}
grid=GridSearchCV(estimator,param_grid,scoring="r2",cv=5)
grid.fit(xtrain,ytrain)
print(grid.best_params_)
rfmodel=RandomForestRegressor(random_state=42,n_estimators=120)
rfmodel.fit(xtrain,ytrain)
ypredtrain2=rfmodel.predict(xtrain)
ypredtest2=rfmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain2))
print(r2_score(ytest,ypredtest2))
print(cross_val_score(rfmodel,X,Y,cv=5,).mean())
```

```
0.9738163260247533
0.8819423353068565
0.8363637309718952
{'n_estimators': 120}
0.9746383984429655
0.8822009842175969
0.8367438097052858
```

```
gbmodel=GradientBoostingRegressor()
gbmodel.fit(xtrain,ytrain)
ypredtrain3=gbmodel.predict(xtrain)
ypredtest3=gbmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain3))
print(r2_score(ytest,ypredtest3))
print(cross_val_score(gbmodel,X,Y,cv=5,).mean())
from sklearn.model_selection import GridSearchCV
estimator=GradientBoostingRegressor()
param_grid={'n_estimators':[10,15,19,20,21,50],'learning_rate':[0.1,0.19,0.2,0.21,0.8,1]}
grid=GridSearchCV(estimator,param_grid,scoring="r2",cv=5)
grid.fit(xtrain,ytrain)
print(grid.best_params_)
gbmodel=GradientBoostingRegressor(n_estimators=19,learning_rate=0.2)
gbmodel.fit(xtrain,ytrain)
ypredtrain3=gbmodel.predict(xtrain)
ypredtest3=gbmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain3))
print(r2_score(ytest,ypredtest3))
print(cross_val_score(gbmodel,X,Y,cv=5,).mean())
```

```
0.893134582116604
0.904261922040551
0.8548269934267699
{'learning_rate': 0.2, 'n_estimators': 19}
0.8682397447116927
0.9017109716082661
0.8606041910125791
```

```
xgmodel=XGBRegressor()
xgmodel.fit(xtrain,ytrain)
ypredtrain4=xgmodel.predict(xtrain)
ypredtest4=xgmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain4))
print(r2_score(ytest,ypredtest4))
print(cross_val_score(xgmodel,X,Y,cv=5,).mean())
from sklearn.model_selection import GridSearchCV
```

```
estimator=XGBRegressor()
param_grid={'n_estimators':[10,15,20,40,50],'max_depth':[3,4,5],'gamma':[0,0.15,0.3,0.5,1]}
grid=GridSearchCV(estimator,param_grid,scoring="r2",cv=5)
grid.fit(xtrain,ytrain)
print(grid.best_params_)
xgmodel=XGBRegressor(n_estimators=15,max_depth=3,gamma=0)
xgmodel.fit(xtrain,ytrain)
ypredtrain4=xgmodel.predict(xtrain)
ypredtest4=xgmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain4))
print(r2_score(ytest,ypredtest4))
print(cross_val_score(xgmodel,X,Y,cv=5,).mean())
```

```
0.9954123497078247
0.8548937785039912
0.808125309217053
{'gamma': 0, 'max_depth': 3, 'n_estimators': 10}
0.8693173313051628
0.9022460881213404
0.8607115291219747
```

```
feats=pd.DataFrame(data=grid.best_estimator_.feature_importances_,index=X.columns,columns=['Importance'])
feats
```

	Importance
age	0.038633
sex	0.000000
bmi	0.133449
children	0.011073
smoker	0.809626
region	0.007219

```
important_features=feats[feats['Importance']>0.01]
important_features
```

	Importance
age	0.038633
bmi	0.133449
children	0.011073
smoker	0.809626

```
df.drop(df[['sex','region']],axis=1,inplace=True)
Xf=df.drop(df[['charges']],axis=1)
X=df.drop(df[['charges']],axis=1)
xtrain,xtest,ytrain,ytest=train_test_split(Xf,Y,test_size=0.2,random_state=42)
finalmodel=XGBRegressor(n_estimators=15,max_depth=3,gamma=0)
finalmodel.fit(xtrain,ytrain)
ypredtrain4=finalmodel.predict(xtrain)
ypredtest4=finalmodel.predict(xtest)
print(r2_score(ytrain,ypredtrain4))
print(r2_score(ytest,ypredtest4))
print(cross_val_score(finalmodel,X,Y,cv=5,).mean())
```

```
0.869105118970057
0.9007425513499979
0.8606266871712276
```

```
from pickle import dump
```

```
dump(finalmodel,open('insurancemodelf.pkl','wb'))
```

```
new_data=pd.DataFrame({'age':19,'sex':'male','bmi':27.9,'children':0,'smoker':'yes','region':'northeast'},index=[0])
new_data['smoker']=new_data['smoker'].map({'yes':1,'no':0})
new_data=new_data.drop(new_data[['sex','region']],axis=1)
finalmodel.predict(new_data)
```

```
array([18035.828], dtype=float32)
```

```
"""Conclusion
Out of all the models XGBoost model is giving the highest accuracy this means predictions made by this model are close to the real va

The dataset we have used here was small still the conclusion we drew from them were quite similar to what is observed in the real-lif

"""
```

```
'Conclusion\nOut of all the models XGBoost model is giving the highest accuracy this means predictions made by this model are c
lose to the real values as compared to the other model.\n\nThe dataset we have used here was small still the conclusion we drew
from them were quite similar to what is observed in the real-life scenario. If we would have a bigger dataset then we will be a
ble to learn even deeper patterns in the relation between the independent features and the premium charged from the buyers.\n
\n'
```