

File Edit View Run Add-ons Help

+ Run All | Code ▾

```
[99]: # This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

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

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

/kaggle/input/heart-disease-prediction-using-logistic-regression/framingham.csv
```

```
[100]: df = pd.read_csv('/kaggle/input/heart-disease-prediction-using-logistic-regression/framingham.csv')
```

```
[101]: df.head(10)
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0
5	0	43	2.0	0	0.0	0.0	0	1	0	228.0	180.0	110.0	30.30	77.0	99.0	0
6	0	63	1.0	0	0.0	0.0	0	0	0	205.0	138.0	71.0	33.11	60.0	85.0	1
7	0	45	2.0	1	20.0	0.0	0	0	0	313.0	100.0	71.0	21.68	79.0	78.0	0
8	1	52	1.0	0	0.0	0.0	0	1	0	260.0	141.5	89.0	26.36	76.0	79.0	0
9	1	43	1.0	1	30.0	0.0	0	1	0	225.0	162.0	107.0	23.61	93.0	88.0	0

```
[102]: df.shape
```

```
[10]: (4238, 16)
```

```
[103]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   male             4238 non-null   int64  
 1   age              4238 non-null   int64  
 2   education        4133 non-null   float64 
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64 
 5   BPMeds           4185 non-null   float64 
 6   prevalentStroke  4238 non-null   int64  
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes          4238 non-null   int64  
 9   totChol           4188 non-null   float64 
 10  sysBP             4238 non-null   float64 
 11  diaBP             4238 non-null   float64 
 12  BMI               4219 non-null   float64 
 13  heartRate         4238 non-null   float64 
 14  glucose            3850 non-null   float64 
 15  TenYearCHD        4238 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

```
[104]: df.isnull().sum()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevale	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
male	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
age	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
education	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
currentSmoker	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
cigsPerDay	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
BPMeds	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
prevale	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
prevalentHyp	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
diabetes	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
totChol	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
sysBP	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
diaBP	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
BMI	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
heartRate	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
glucose	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0
TenYearCHD	0	0	105	0	29	53	0	0	0	50	0	0	19	1	0	0

```
[105]: total_null = df.isnull().sum().sort_values(ascending = False)
percent = ((df.isnull().sum()/df.isnull().count())*100).sort_values(ascending = False)
print("Total records = ", df.shape[0])
```

```
missing_data = pd.concat([total_null,percent.round(2)],axis=1,keys=['Total Missing','In Percent'])
```

```
missing_data
```

```
Total records = 4238
```

	Total Missing	In Percent
--	---------------	------------

	Total Missing	In Percent
Glucose	388	9.16
education	105	2.48
BPMeds	53	1.25
totChol	50	1.18
cigsPerDay	29	0.68
BMI	19	0.45
heartRate	1	0.02
male	0	0.00
age	0	0.00

currentSmoker	0	0.00
prevalentStroke	0	0.00
prevalentHyp	0	0.00
diabetes	0	0.00
sysBP	0	0.00
diaBP	0	0.00
TenYearCHD	0	0.00

As the percentage of null values is not very high we will replace the null values

```
[106]: df['glucose'].fillna(value = df['glucose'].mean().round(0), inplace=True)
df['education'].fillna(value = df['education'].mean().round(0), inplace=True)
df['BPMed'].fillna(value = df['BPMed'].median(), inplace=True)
df['totChol'].fillna(value = df['totChol'].median(), inplace=True)
df['BPMed'].fillna(value = df['BPMed'].median(), inplace=True)
df['cigsPerDay'].fillna(value = df['cigsPerDay'].median(), inplace=True)
df['BMI'].fillna(value = df['BMI'].median(), inplace=True)
df['heartRate'].fillna(value = df['heartRate'].median(), inplace=True)
```

```
[107]: df.isnull().sum()
```

```
[108]: male      0
age       0
education 0
currentSmoker 0
cigsPerDay 0
BPMed     0
prevalentStroke 0
prevalentHyp 0
diabetes   0
totChol    0
sysBP     0
diaBP     0
BMI       0
heartRate 0
glucose   0
TenYearCHD 0
dtype: int64
```

No null values(Data is cleaned)

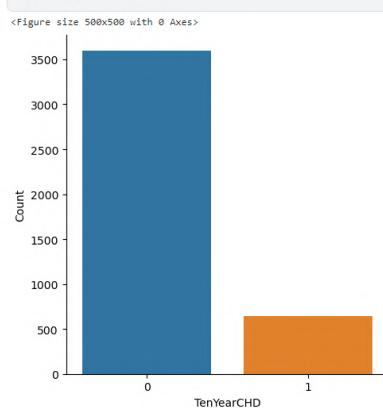
```
[109]: df.describe()
```

```
[109]:    male      age education currentSmoker cigsPerDay BPMed prevalentStroke prevalentHyp diabetes totChol sysBP diaBP BMI heartRate glucose TenYearCHD
count 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000 4238.000000
mean  0.429212 49.584946 1.979471 0.494101 8.941482 0.029259 0.005899 0.310524 0.025720 236.689476 132.352407 82.893464 25.800205 75.878716 81.969797 0.151958
std   0.495022 8.572160 1.007081 0.500024 11.902399 0.168552 0.076587 0.462763 0.158316 44.327427 22.038097 11.910850 4.071041 12.025185 22.836605 0.359023
min   0.000000 32.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 107.000000 83.500000 48.000000 15.540000 44.000000 40.000000 0.000000
25%  0.000000 42.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 206.000000 117.000000 75.000000 23.080000 68.000000 72.000000 0.000000
50%  0.000000 49.000000 2.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 234.000000 128.000000 82.000000 25.400000 75.000000 80.000000 0.000000
75%  1.000000 56.000000 3.000000 1.000000 20.000000 0.000000 1.000000 0.000000 262.000000 144.000000 89.875000 280.37500 83.000000 85.000000 0.000000
max  1.000000 70.000000 4.000000 1.000000 70.000000 1.000000 1.000000 1.000000 1.000000 696.000000 295.000000 142.500000 56.800000 143.000000 394.000000 1.000000
```

Importing Libraries

```
[109]: import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
[110]: plt.figure(figsize=(5,5))
sns.catplot(x="TenYearCHD", data = df, kind = 'count')
plt.xlabel('TenYearCHD')
plt.ylabel('Count')
plt.show()
```



```
[111]: correlation = df.corr()
```

```
[112]: print(correlation[['TenYearCHD']])
```

	TenYearCHD
male	0.088428
age	0.225256
education	-0.053383
currentSmoker	0.019456
cigsPerDay	0.058859
BPMed	0.057072
prevalentStroke	0.061810
prevalentHyp	0.177603
diabetes	0.097317
totChol	0.081566
sysBP	0.216429
diaBP	0.145290
BMI	0.091717
heartRate	0.022857
glucose	0.120398
TenYearCHD	1.000000
Name:	TenYearCHD, dtype: float64

From the correlation matrix TeNYearCHD is mostly related to age and sysBP

Model Building

Training the Machine Learning Model

```
[113]: X = df.drop("TenYearCHD", axis="columns")
```

[114]

[11]	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0
...
4233	1	50	1.0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.97	66.0	86.0
4234	1	51	3.0	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19.71	65.0	68.0
4235	0	48	2.0	1	20.0	0.0	0	0	0	248.0	131.0	72.0	22.00	84.0	86.0
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	86.0	82.0
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	80.0	107.0

4238 rows × 15 columns

```
[115]: Y = df['TenYearCHD']
```

[116]

```
[11] 0 0  
    1 0  
    2 0  
    3 1  
    4 0  
    ..  
4233 1  
4234 0  
4235 0  
4236 0  
4237 0  
Name: TenYearCHD, Length: 4238, dtype: int64
```

```
[117]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
[118]: model = LogisticRegression()
```

```
[1]: %%capture  
import warnings  
warnings.filterwarnings('ignore')  
  
#opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):  
    #TOTAL NO. OF ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max_iter) or scale the data as shown:  
    https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression  
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

(3300, 15)

```
[121]: X_test_prediction = lr.predict(X_test)
Y_test_prediction
```

```
[122]: accuracy_score(X_test_prediction, Y_test)
```

[12... 0.8549528301886793

```
[123]: lrscore = lr.score(X_test, Y_test)  
lrscore
```

[12... 0.8549528301886793

```
[124]: from sklearn import tree
```

```
[125]: clf = tree.DecisionTreeClassifier()
```

```
[126]: clf = clf.fit(X_train,Y_train)
```

```
[127]: accuracyscoretree = clf.score(accuracytree
```

[12] 0.7653301886792453

[128]: `treee.plot_tree`


```

Text(0.8263091786099591, '0.455172413779310345', 'gini = 0.0\nsamples = 6\nvalue = [6, 0]',  

Text(0.8290061107365636, '0.109655172417931', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.83427999007725, '0.258620689517243', 'X[1] < 61.5\ngini = 0.444\nsamples = 6\nvalue = [4, 2]',  

Text(0.83165205872168, '0.2213793103448276', 'gini = 0.0\nsamples = 2\nvalue = [0, 2]',  

Text(0.8369439391437368, '0.22413793103448276', 'gini = 0.0\nsamples = 17\nvalue = [17, 0]',  

Text(0.8369439391437368, '0.3636889317714', 'X[12] < 27.945\ngini = 0.5\nsamples = 2\nvalue = [1, 1]',  

Text(0.8369439391437368, '0.3758620689517243', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.83598987289893, '0.32758620689517243', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.8369439391437368, '0.39655172413793105', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.8448817595801902, '0.43103448275862066', 'X[1] < 63.5\ngini = 0.043\nsamples = 46\nvalue = [45, 1]',  

Text(0.842235194145857, '0.39655172413793105', 'gini = 0.0\nsamples = 40\nvalue = [40, 0]',  

Text(0.8448817595801902, '0.36208893172414', 'gini = 0.0\nsamples = 1\nvalue = [5, 0]',  

Text(0.8448817595801902, '0.36208893172414', 'gini = 0.0\nsamples = 1\nvalue = [0, 5]',  

Text(0.8528195799570035, '0.46551724137931033', 'X[10] < 183.5\ngini = 0.444\nsamples = 6\nvalue = [4, 2]',  

Text(0.8501736398213998, '0.43103448275862066', 'gini = 0.0\nsamples = 2\nvalue = [0, 2]',  

Text(0.8546455208926079, '0.43103448275862066', 'gini = 0.0\nsamples = 4\nvalue = [4, 0]',  

Text(0.86340833404994212, '0.5344827586206896', 'X[10] < 172.5\ngini = 0.469\nsamples = 8\nvalue = [5, 3]',  

Text(0.869705194145857, '0.5344827586206896', 'X[10] < 17.5\ngini = 0.278\nsamples = 6\nvalue = [5, 1]',  

Text(0.869705194145857, '0.5344827586206896', 'X[10] < 17.5\ngini = 0.278\nsamples = 6\nvalue = [5, 1]',  

Text(0.86340033404994212, '0.46551724137931033', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.8660492806350256, '0.5689655172413793', 'gini = 0.0\nsamples = 2\nvalue = [0, 2]',  

Text(0.8544732925417562, '0.5689655172413793', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.862411112485695, '0.6724137931034483', 'X[5] < 0.5\ngini = 0.486\nsamples = 12\nvalue = [7, 5]',  

Text(0.862411112485695, '0.6724137931034483', 'X[10] < 170.5\ngini = 0.486\nsamples = 10\nvalue = [7, 3]',  

Text(0.877119232773608, '0.69344827586206896', 'X[10] < 183.75\ngini = 0.5\nsamples = 6\nvalue = [3, 3]',  

Text(0.862411112485695, '0.5689655172413793', 'gini = 0.0\nsamples = 3\nvalue = [3, 0]',  

Text(0.8597651728129652, '0.5689655172413793', 'gini = 0.0\nsamples = 3\nvalue = [0, 3]',  

Text(0.865857053084174, '0.6379310344827587', 'gini = 0.0\nsamples = 2\nvalue = [0, 2]',  

Text(0.8703489333553829, '0.77586206895172413793', 'X[12] < 41.09\ngini = 0.497\nsamples = 13\nvalue = [6, 7]',  

Text(0.8703489333553829, '0.77586206895172413793', 'X[12] < 41.09\ngini = 0.497\nsamples = 13\nvalue = [1, 6]',  

Text(0.862411112485695, '0.76689655172413793', 'gini = 0.0\nsamples = 1\nvalue = [1, 0]',  

Text(0.8677029932197784, '0.76689655172413793', 'gini = 0.0\nsamples = 6\nvalue = [0, 6]',  

Text(0.87564801362595172413793, '0.741379310344827587', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.87299487340990873, '0.70688655172413793', 'gini = 0.0\nsamples = 5\nvalue = [5, 0]',  

Text(0.872867537621961, '0.70688655172413793', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.8808750510344827587, '0.844827586206895172413793', 'X[1] < 56.0\ngini = 0.489\nsamples = 31\nvalue = [15, 18]',  

Text(0.8808750510344827587, '0.844827586206895172413793', 'X[1] < 56.0\ngini = 0.489\nsamples = 31\nvalue = [15, 18]',  

Text(0.8809126938978005, '0.77586206895172413793', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.8809126938978005, '0.77586206895172413793', 'gini = 0.0\nsamples = 5\nvalue = [5, 0]',  

Text(0.8862245741690095, '0.77586206895172413793', 'gini = 0.0\nsamples = 5\nvalue = [5, 0]',  

Text(0.8941623945758227, '0.8103448275862069', 'X[12] < 41.545\ngini = 0.435\nsamples = 25\nvalue = [8, 17]',  

Text(0.8915164544021883, '0.77586206895172413793', 'X[11] < 104.75\ngini = 0.386\nsamples = 23\nvalue = [6, 17]',  

Text(0.8925245741690095, '0.741379310344827587', 'X[10] < 79.5\ngini = 0.5\nsamples = 10\nvalue = [5, 5]',  

Text(0.88802460334085, '0.77586206895172413793', 'X[10] < 180.0\ngini = 0.278\nsamples = 20\nvalue = [2, 5]',  

Text(0.8809126938978005, '0.6724137931034483', 'X[12] < 25.995\ngini = 0.444\nsamples = 3\nvalue = [2, 1]',  

Text(0.8782867537621961, '0.67241379310344827587', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.883578634033405, '0.6379310344827587', 'gini = 0.0\nsamples = 4\nvalue = [0, 4]',  

Text(0.8888705143046138, '0.70688655172413793', 'gini = 0.0\nsamples = 3\nvalue = [3, 0]',  

Text(0.8888705143046138, '0.70688655172413793', 'X[14] < 12.0\ngini = 0.444\nsamples = 13\nvalue = [1, 12]',  

Text(0.8941623945758227, '0.66089655172413793', 'gini = 0.0\nsamples = 1\nvalue = [1, 0]',  

Text(0.894542748470316, '0.70688655172413793', 'gini = 0.0\nsamples = 12\nvalue = [0, 12]',  

Text(0.8955878948238793, '0.8793103448275826', 'X[9] < 200.0\ngini = 0.5\nsamples = 124\nvalue = [61, 63]',  

Text(0.9100830353894493, '0.844827586206895172413793', 'X[12] < 24.94\ngini = 0.375\nsamples = 28\nvalue = [21, 7]',  

Text(0.90474515511183094, '0.8103448275862069', 'X[10] < 180.0\ngini = 0.486\nsamples = 9\nvalue = [4, 5]',  

Text(0.9073020525384494, '0.77586206895172413793', 'X[13] < 85.0\ngini = 0.321\nsamples = 5\nvalue = [0, 4]',  

Text(0.90474615511182404, '0.741379310344827587', 'gini = 0.0\nsamples = 4\nvalue = [4, 0]',  

Text(0.90474615511182404, '0.741379310344827587', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.9153299156606581, '0.8103448275862069', 'X[10] < 104.75\ngini = 0.188\nsamples = 19\nvalue = [17, 2]',  

Text(0.91268397552595172413793, '0.77586206895172413793', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.9153299156606581, '0.77586206895172413793', 'X[12] < 180.0\ngini = 0.486\nsamples = 18\nvalue = [17, 1]',  

Text(0.9153299156606581, '0.741379310344827587', 'gini = 0.0\nsamples = 16\nvalue = [16, 0]',  

Text(0.920621759531867, '0.741379310344827587', 'X[12] < 29.125\ngini = 0.5\nsamples = 3\nvalue = [1, 1]',  

Text(0.9179758557962626, '0.70688655172413793', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.920621759531867, '0.70688655172413793', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.9611377542583099, '0.844827586206895172413793', 'X[14] < 71.5\ngini = 0.486\nsamples = 96\nvalue = [40, 56]',  

Text(0.9353337683569831, '0.8103448275862069', 'X[11] < 52.0\ngini = 0.435\nsamples = 25\nvalue = [17, 8]',  

Text(0.9353337683569831, '0.8103448275862069', 'X[11] < 104.75\ngini = 0.435\nsamples = 8\nvalue = [3, 5]',  

Text(0.9312095564742844, '0.741379310344827587', 'X[11] < 103.5\ngini = 0.375\nsamples = 17\nvalue = [3, 1]',  

Text(0.9285596136386983, '0.741379310344827587', 'gini = 0.0\nsamples = 1\nvalue = [0, 1]',  

Text(0.9338514966098892, '0.70688655172413793', 'gini = 0.0\nsamples = 3\nvalue = [3, 0]',  

Text(0.9364974367454937, '0.741379310344827587', 'gini = 0.0\nsamples = 4\nvalue = [0, 4]',  

Text(0.9444352571523069, '0.77586206895172413793', 'X[11] < 79.0\ngini = 0.291\nsamples = 17\nvalue = [14, 3]')
...]

```

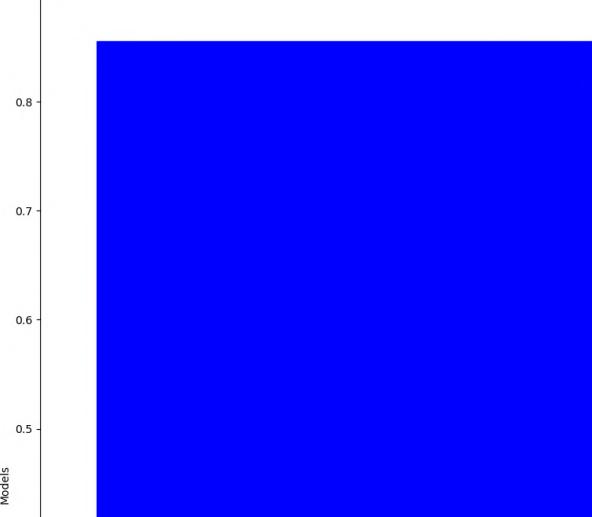


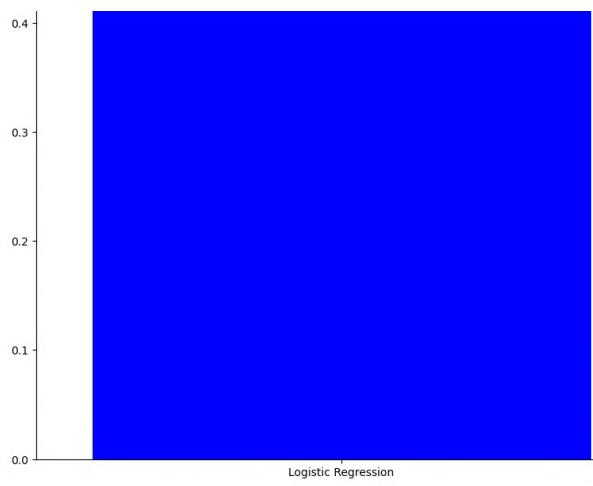
+ Code + Markdown

```

names=['Logistic Regression', 'Decision Tree Classification']
acc=[lrcscore,accuracyscoreter]
plt.figure(figsize=(20,16))
graph = plt.bar(names,acc)
plt.xlabel('Accuracy')
plt.ylabel('Models')
graph[0].set_color('blue')
graph[1].set_color('grey')

```





[+ Code](#) [+ Markdown](#)

