In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix,auc,roc_auc_score,precision_score,recall_score
```

In [2]:
```
df = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

In [4]:
```
df.shape
```

Out[4]:
```
(5110, 12)
```

In [5]:
```
df.describe()
```

Out[5]:

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54932.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

Data Preprocessing

Data Cleaning

Missing value for Train and Test Data

In [6]:
```
df.isnull().sum()/len(df)*100
```

Out[6]:
```
id                   0.000000
gender               0.000000
age                  0.000000
hypertension         0.000000
heart_disease        0.000000
ever_married         0.000000
work_type            0.000000
Residence_type       0.000000
avg_glucose_level    0.000000
bmi                  3.933464
smoking_status       0.000000
stroke               0.000000
dtype: float64
```

So, Data has bmi 3.933% of missing value

In [7]:
```
df["bmi"] = df["bmi"].fillna(df["bmi"].mean())
```

In [8]:
```
df["bmi"]
```

Out[8]:
```
0       36.600000
1       28.893237
2       32.500000
3       34.400000
4       24.000000
        ...
5105    28.893237
5106    40.000000
5107    30.600000
5108    25.600000
5109    26.200000
Name: bmi, Length: 5110, dtype: float64
```

In [9]:
```
label = LabelEncoder()
df['gender']=label.fit_transform(df['gender'])
df['ever_married']=label.fit_transform(df['ever_married'])
df['work_type']=label.fit_transform(df['work_type'])
df['Residence_type']=label.fit_transform(df['Residence_type'])
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

In [11]:
```
df.columns
```

Out[11]:
```
Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

In [12]:
```
#splitting
```

In [13]:
```
X = df.drop(['id',"gender",'age','hypertension','heart_disease','ever_married',"work_type","Residence_type",'avg_glucose_level','bmi',"smoking_status"],axis=1)
y = df['stroke']
```

In [14]:
```
df.head()
```

Out[14]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	1	67.0	0	1	1	2	1	228.69	36.600000	formerly smoked	1
1	51676	0	61.0	0	0	1	3	0	202.21	28.893237	never smoked	1
2	31112	1	80.0	0	1	1	2	0	105.92	32.500000	never smoked	1
3	60182	0	49.0	0	0	1	2	1	171.23	34.400000	smokes	1
4	1665	0	79.0	1	0	1	3	0	174.12	24.000000	never smoked	1

In [15]:
```
X_shape
```

Out[15]:
```
(5110, 1)
```

In [16]:
```
y_shape
```

Out[16]:
```
(5110,)
```

In [17]:
```
df['stroke'].value_counts()
```

Out[17]:
```
0    4861
1     249
Name: stroke, dtype: int64
```

In [18]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [19]:
```
scaler = StandardScaler()
```

In [21]:
```
X_scaled = scaler.fit_transform(X)
```

In [22]:
```
#splitting

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.20, random_state=0)
```

In [23]:
```
# import RF
```

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [25]:
```
classifier = RandomForestClassifier(n_estimators=100)
```

In [26]:
```
classifier.fit(X_train,y_train)
```

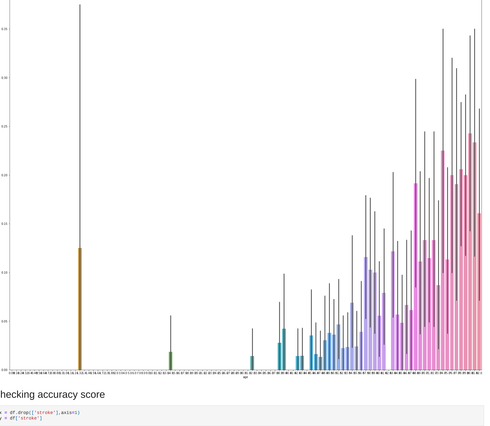
Out[26]:
```
RandomForestClassifier()
```

In [27]:
```
y_pred = classifier.predict(X_test)
```

In [28]:
```
print('Accuracy:',accuracy_score(y_pred,y_test))
```

```
Accuracy: 1.0
```

In [29]:
```
plt.figure(figsize=(30,25))
sns.barplot(df["age"],df["stroke"])
plt.show()
```



checking accuracy score

In [30]:
```
x = df.drop(['stroke'],axis=1)
y = df['stroke']
```

In [31]:
```
#splitting

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33, random_state=42)
```

In [32]:
```
RFC = RandomForestClassifier(random_state=150)
```

In [33]:
```
RFC.fit(X_train,y_train)
```

Out[33]:
```
RandomForestClassifier(random_state=150)
```

In [34]:
```
y_pread = RFC.predict(X_test)
```

In [35]:
```
print('model accuracy score with 1000decision tree : {0:0.4f}'.format(accuracy_score(y_test,y_pread)))
```

```
model accuracy score with 1000decision tree : 1.0000
```

LogisticRegression

In [37]:
```
from sklearn.linear_model import LogisticRegression
```

In [38]:
```
classifier = LogisticRegression()
```

In [39]:
```
classifier.fit(X_train, y_train)
```

Out[39]:
```
LogisticRegression()
```

In [40]:
```
#evaluate the model
```

In [41]:
```
ypred = classifier.predict(X_test)
```

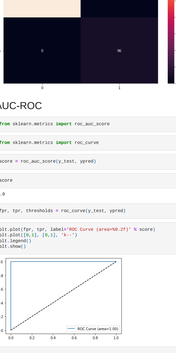
In [42]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

In [43]:
```
cm = confusion_matrix(y_test, ypred)
```

In [44]:
```
cm
```

Out[44]:
```
array([[1591,    0],
       [   0,   98]], dtype=int64)
```

In [45]:
```
fig, ax = plt.subplots(figsize=(10,7))
sns.heatmap(cm, annot=True)
plt.show()
```



AUC-ROC

In [46]:
```
from sklearn.metrics import roc_auc_score
```

In [47]:
```
from sklearn.metrics import roc_curve
```

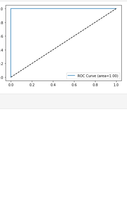
In [48]:
```
score = roc_auc_score(y_test, ypred)
```

In [49]:
```
score
```

Out[49]:
```
1.0
```

In [50]:
```
fpr, tpr, thresholds = roc_curve(y_test, ypred)
```

In [73]:
```
plt.plot(fpr, tpr, label='ROC Curve (area=%0.2f)' % score)
plt.plot([0,1], [0,1], 'k--')
plt.legend()
plt.show()
```



In [ ]:
```

```