In [1]:

**import numpy as np import pandas as pd import os**

**for** dirname, \_, filenames **in** os.walk('D:\stroke\healthcare-dataset-stroke-data.csv'):

**for** filename **in** filenames:

print(os.path.join(dirname, filename))

In [2]:

**import matplotlib.pyplot as plt import seaborn as sns**

**import warnings**

warnings.filterwarnings('ignore')

In [3]:

df = pd.read\_csv('D:\stroke\healthcare-dataset-stroke-data.csv') df.head(10)

Out[3]:

**id gender age hypertension heart\_disease ever\_married work\_type Residence\_ty**

1. 9046 Male 67.0 0 1 Yes Private Urb
2. 51676 Female 61.0 0 0 Yes Self- Ru

employed

1. 31112 Male 80.0 0 1 Yes Private Ru
2. 60182 Female 49.0 0 0 Yes Private Urb
3. 1665 Female 79.0 1 0 Yes Self- Ru

employed

1. 56669 Male 81.0 0 0 Yes Private Urb
2. 53882 Male 74.0 1 1 Yes Private Ru
3. 10434 Female 69.0 0 0 No Private Urb
4. 27419 Female 59.0 0 0 Yes Private Ru
5. 60491 Female 78.0 0 0 Yes Private Urb

In [4]:

print(df.columns.tolist())

['id', 'gender', 'age', 'hypertension', 'heart\_disease', 'ever\_married', 'work\_type', 'Residence\_type', 'avg\_glucose\_level', 'bmi', 'smoking\_statu s', 'stroke']

df.shape

*# find how many data we have and how many type of data # 5110 rows = total data , 12 columns = 12 types*

Out[5]:

(5110, 12)

In [6]:

df.info()

*# check whether there are null value*

<class 'pandas.core.frame.DataFrame'> RangeIndex: 5110 entries, 0 to 5109

Data columns (total 12 columns):

# Column Non-Null Count Dtype

1. id 5110 non-null int64
2. gender 5110 non-null object
3. age 5110 non-null float64
4. hypertension 5110 non-null int64
5. heart\_disease 5110 non-null int64
6. ever\_married 5110 non-null object
7. work\_type 5110 non-null object
8. Residence\_type 5110 non-null object
9. avg\_glucose\_level 5110 non-null float64
10. bmi 4909 non-null float64
11. smoking\_status 5110 non-null object
12. stroke 5110 non-null int64 dtypes: float64(3), int64(4), object(5)

memory usage: 479.2+ KB

In [7]:

data = df.drop('id', axis=1) *# id is useless, so we drop it*

data.isnull().sum() *# find how many null value in bmi*

|  |  |
| --- | --- |
| Out[7]: |  |
| gender | 0 |
| age | 0 |
| hypertension | 0 |
| heart\_disease | 0 |
| ever\_married | 0 |
| work\_type | 0 |
| Residence\_type | 0 |
| avg\_glucose\_level | 0 |
| bmi | 201 |
| smoking\_status | 0 |
| stroke  dtype: int64 | 0 |
| In [8]: |  |

data['bmi'].fillna(data['bmi'].mean(), inplace=**True**)

*# replace null value by mean*

data.isnull().sum()

Out[9]:

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 [10]:

features = ['gender', 'hypertension', 'heart\_disease','ever\_married', 'work\_type', 'Res idence\_type', 'smoking\_status']

**for** feature **in** features:

print(data[feature].unique())

*# find unique value*

*# like in gender: Other(only 1, so ignore)*

*# like in smoking\_status: unknown(looks like there are a lot, I used mode to replace th em)*

['Male' 'Female' 'Other'] [0 1]

[1 0]

['Yes' 'No']

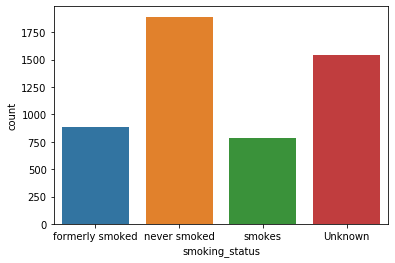
['Private' 'Self-employed' 'Govt\_job' 'children' 'Never\_worked'] ['Urban' 'Rural']

['formerly smoked' 'never smoked' 'smokes' 'Unknown']

print(data.smoking\_status.value\_counts()) sns.countplot(x = data['smoking\_status']) plt.show()

|  |  |
| --- | --- |
| never smoked | 1892 |
| Unknown | 1544 |
| formerly smoked | 885 |
| smokes | 789 |

Name: smoking\_status, dtype: int64

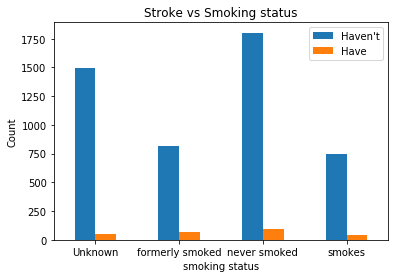


pd.crosstab(data.smoking\_status,data.stroke).plot(kind="bar") plt.title('Stroke vs Smoking status')

plt.xlabel('smoking status') plt.xticks(rotation = 0)

plt.legend(["Haven't", "Have"]) plt.ylabel('Count')

plt.show()



In [13]:

data['smoking\_status'].replace('Unknown', np.nan, inplace=**True**)

data['smoking\_status'].fillna(data['smoking\_status'].mode()[0], inplace = **True**)

data.describe()

Out[14]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **hypertension** | **heart\_disease** | **avg\_glucose\_level** | **bmi** | **strok** |
| **count** | 5110.000000 | 5110.000000 | 5110.000000 | 5110.000000 | 5110.000000 | 5110.00000 |
| **mean** | 43.226614 | 0.097456 | 0.054012 | 106.147677 | 28.893237 | 0.04872 |
| **std** | 22.612647 | 0.296607 | 0.226063 | 45.283560 | 7.698018 | 0.21532 |
| **min** | 0.080000 | 0.000000 | 0.000000 | 55.120000 | 10.300000 | 0.00000 |
| **25%** | 25.000000 | 0.000000 | 0.000000 | 77.245000 | 23.800000 | 0.00000 |
| **50%** | 45.000000 | 0.000000 | 0.000000 | 91.885000 | 28.400000 | 0.00000 |
| **75%** | 61.000000 | 0.000000 | 0.000000 | 114.090000 | 32.800000 | 0.00000 |
| **max** | 82.000000 | 1.000000 | 1.000000 | 271.740000 | 97.600000 | 1.00000 |

In [15]:

data.corr()

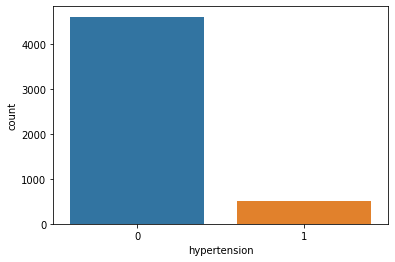
*# varibales by varibles*

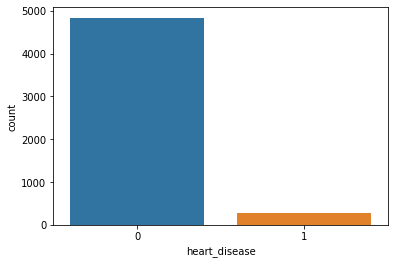
Out[15]:

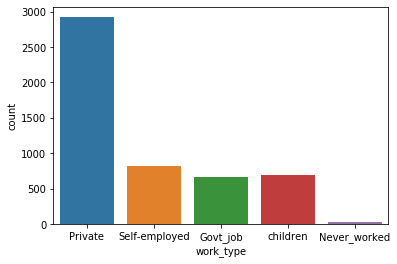
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **hypertension** | **heart\_disease** | **avg\_glucose\_level** | **bmi** | **str** |
| **age** | 1.000000 | 0.276398 | 0.263796 | 0.238171 | 0.325942 | 0.245 |
| **hypertension** | 0.276398 | 1.000000 | 0.108306 | 0.174474 | 0.160189 | 0.127 |
| **heart\_disease** | 0.263796 | 0.108306 | 1.000000 | 0.161857 | 0.038899 | 0.134 |
| **avg\_glucose\_level** | 0.238171 | 0.174474 | 0.161857 | 1.000000 | 0.168751 | 0.131 |
| **bmi** | 0.325942 | 0.160189 | 0.038899 | 0.168751 | 1.000000 | 0.038 |
| **stroke** | 0.245257 | 0.127904 | 0.134914 | 0.131945 | 0.038947 | 1.000 |

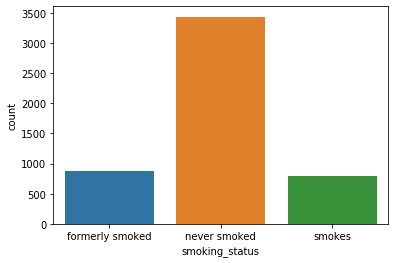
**for** feature **in** features: plt.figure()

sns.countplot(x = data[feature]) plt.show()









print(data.hypertension.value\_counts()) print(" ")

print(data.heart\_disease.value\_counts()) print(" ")

print(data.ever\_married.value\_counts()) print(" ")

print(data.gender.value\_counts()) print(" ")

print(data.work\_type.value\_counts()) print(" ")

print(data.Residence\_type.value\_counts()) print(" ")

print(data.smoking\_status.value\_counts()) print(" ")

print(data.stroke.value\_counts())

0 4612

1 498

Name: hypertension, dtype: int64

0 4834

1 276

Name: heart\_disease, dtype: int64

Yes 3353

No 1757

Name: ever\_married, dtype: int64

|  |  |  |  |
| --- | --- | --- | --- |
| Female | 2994 |  | |
| Male | 2115 |
| Other  Name: | 1  gender, | dtype: int64 | |
| Private | | | 2925 |
| Self-employed | | | 819 |
| children | | | 687 |
| Govt\_job | | | 657 |
| Never\_worked | | | 22 |
| Name: work\_type, | | | dtype: int64 |

Urban 2596

Rural 2514

Name: Residence\_type, dtype: int64

never smoked 3436

formerly smoked 885

smokes 789

Name: smoking\_status, dtype: int64

0 4861

1 249

Name: stroke, dtype: int64

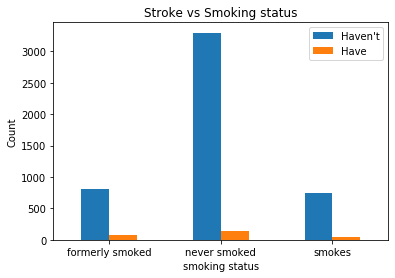
pd.crosstab(data.smoking\_status,data.stroke).plot(kind="bar") plt.title('Stroke vs Smoking status')

plt.xlabel('smoking status') plt.xticks(rotation = 0)

plt.legend(["Haven't", "Have"]) plt.ylabel('Count')

plt.show()

*#we can see smoking status is not relevant to stroke*



In [19]:

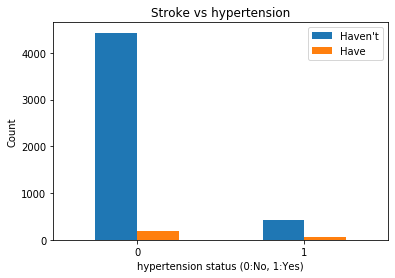
pd.crosstab(data.hypertension,data.stroke).plot(kind="bar") plt.title('Stroke vs hypertension')

plt.xlabel('hypertension status (0:No, 1:Yes)') plt.xticks(rotation = 0)

plt.legend(["Haven't", "Have"]) plt.ylabel('Count')

plt.show()

*#we can see shypertension is not relevant to stroke*

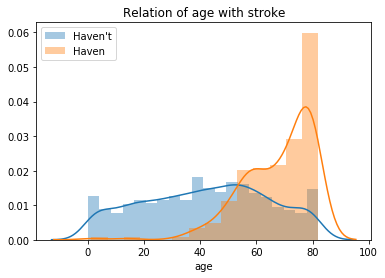


plt.title('Relation of age with stroke') sns.distplot(data.age[data.stroke==0])

sns.distplot(data.age[data.stroke==1]) plt.legend(["Haven't","Haven"])

plt.show()

*#looks like elderly will easily get stroke*



In [21]:

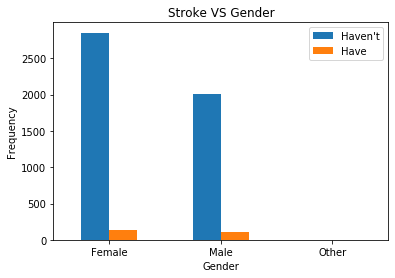
pd.crosstab(data.gender,data.stroke).plot(kind="bar") plt.title('Stroke VS Gender')

plt.xlabel('Gender')

plt.xticks(rotation = 0)

plt.legend(["Haven't", "Have"]) plt.ylabel('Frequency')

plt.show()

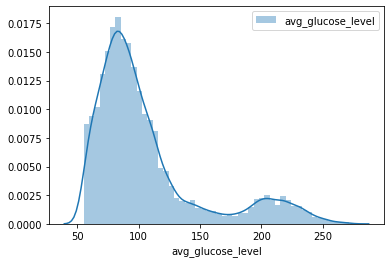


plt.figure()

sns.distplot(data["avg\_glucose\_level"], label="avg\_glucose\_level") plt.legend()

Out[22]:

<matplotlib.legend.Legend at 0x208676f9448>

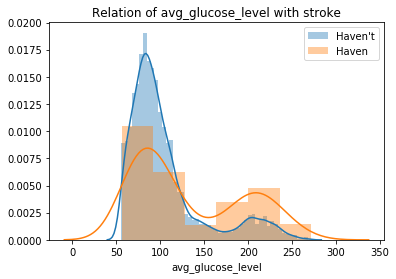


In [23]:

plt.title('Relation of avg\_glucose\_level with stroke') sns.distplot(data.avg\_glucose\_level[data.stroke==0])

sns.distplot(data.avg\_glucose\_level[data.stroke==1]) plt.legend(["Haven't","Haven"])

plt.show()

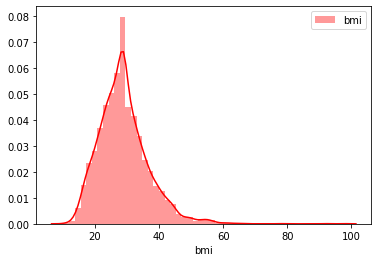


plt.figure()

sns.distplot(data["bmi"], label="bmi",color="red") plt.legend()

Out[24]:

<matplotlib.legend.Legend at 0x208678dabc8>

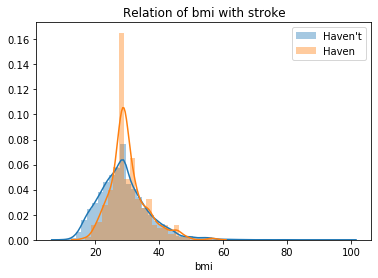


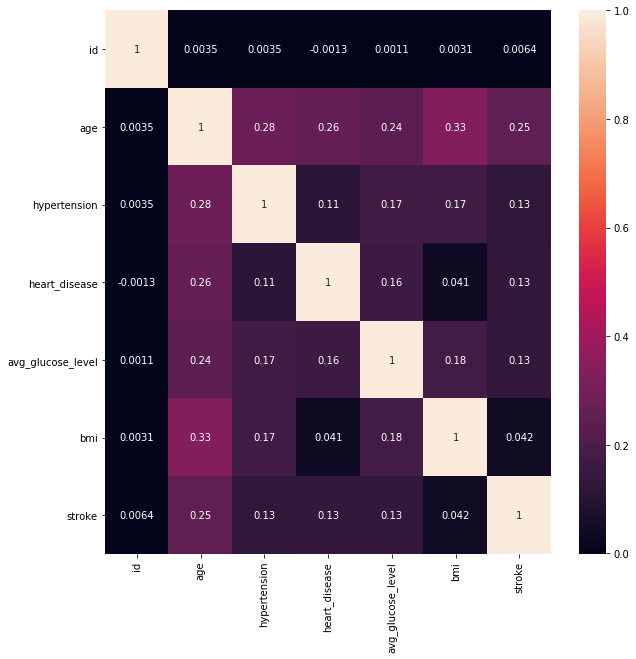
In [25]:

plt.title('Relation of bmi with stroke') sns.distplot(data.bmi[data.stroke==0])

sns.distplot(data.bmi[data.stroke==1]) plt.legend(["Haven't","Haven"])

plt.show()





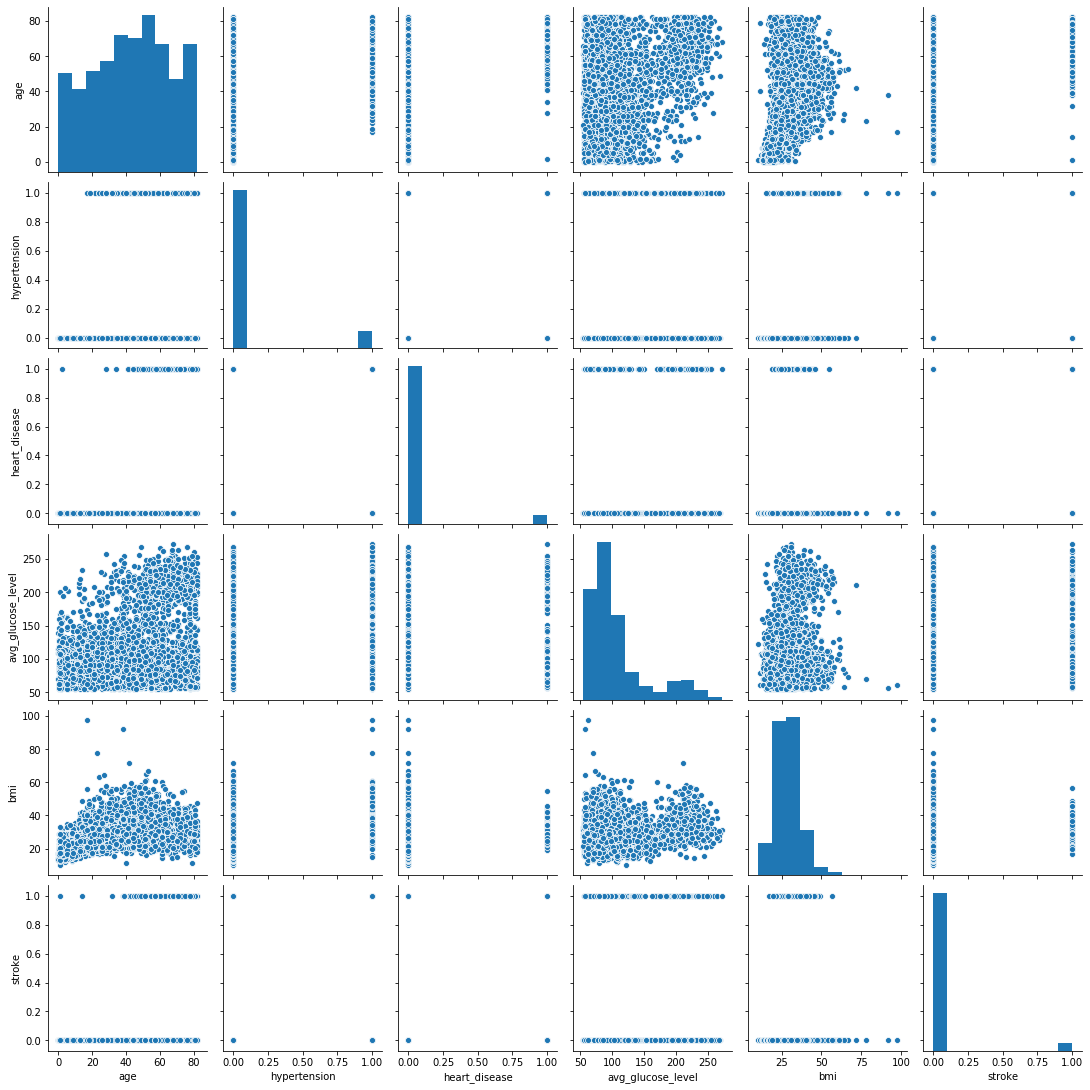
plt.figure(figsize=(10,10))

sns.heatmap(df.corr(),annot=**True**);

plt.figure(figsize=(10,10)) sns.pairplot(data)

plt.show()

<Figure size 720x720 with 0 Axes>



**from sklearn.preprocessing import** LabelEncoder lebalencode = LabelEncoder()

en\_data = data.apply(lebalencode.fit\_transform) en\_data.head()

*# label encoding*

Out[28]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **gender** | **age** | **hypertension** | **heart\_disease** | **ever\_married** | **work\_type** | **Residence\_type** | **avg\_** |
| **0** 1 | 88 | 0 | 1 | 1 | 2 | 1 |  |
| **1** 0 | 82 | 0 | 0 | 1 | 3 | 0 |  |
| **2** 1 | 101 | 0 | 1 | 1 | 2 | 0 |  |
| **3** 0 | 70 | 0 | 0 | 1 | 2 | 1 |  |
| **4** 0 | 100 | 1 | 0 | 1 | 3 | 0 |  |

In [29]:

x = en\_data.drop('stroke', axis = 1) y = en\_data['stroke']

print('X Shape: ', x.shape) print('Y Shape: ', y.shape)

*# feature selection and see now how many rows and columns # after drop stroke, only 10 types data*

X Shape: (5110, 10)

Y Shape: (5110,)

In [30]:

**from sklearn.model\_selection import** train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size= 0.3, random\_state= 0)

*# split data into 30% and 70&*

In [31]:

print("Transactions: ")

print("x\_train: ", x\_train.shape) print("y\_train: ", y\_train.shape) print("x\_test: ", x\_test.shape) print("y\_test: ", y\_test.shape)

*# To see the shape of test dataset and train dataset*

Transactions:

|  |  |
| --- | --- |
| x\_train: | (3577, 10) |
| y\_train: | (3577,) |
| x\_test: | (1533, 10) |
| y\_test: | (1533,) |

**from sklearn.ensemble import** RandomForestClassifier *#for the model*

**from sklearn.tree import** DecisionTreeClassifier *#for the model*

**from sklearn.neighbors import** KNeighborsClassifier *#for the model*

**from sklearn.linear\_model import** LogisticRegression *#for the model*

**from sklearn.metrics import** roc\_curve, auc *#for model evaluation*

**from sklearn.metrics import** classification\_report *#for model evaluation*

**from sklearn.metrics import** confusion\_matrix *#for model evaluation*

**from sklearn.model\_selection import** train\_test\_split *#for data splitting*

In [33]:

*# Evaluation with ROC curve*

**def** plot\_roc(y\_test,y\_proba, model\_name):

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba) fnr = 1 - tpr

eer\_idx = np.nanargmin(np.absolute((fnr - fpr))) eer = (fpr[eer\_idx] + fnr[eer\_idx]) / 2

area = auc(fpr, tpr)

label="**{}** AUC:**{:.2f}** EER:**{:.2f}**%".format(model\_name,area,eer \* 100) plt.plot(fpr, tpr, label=label)

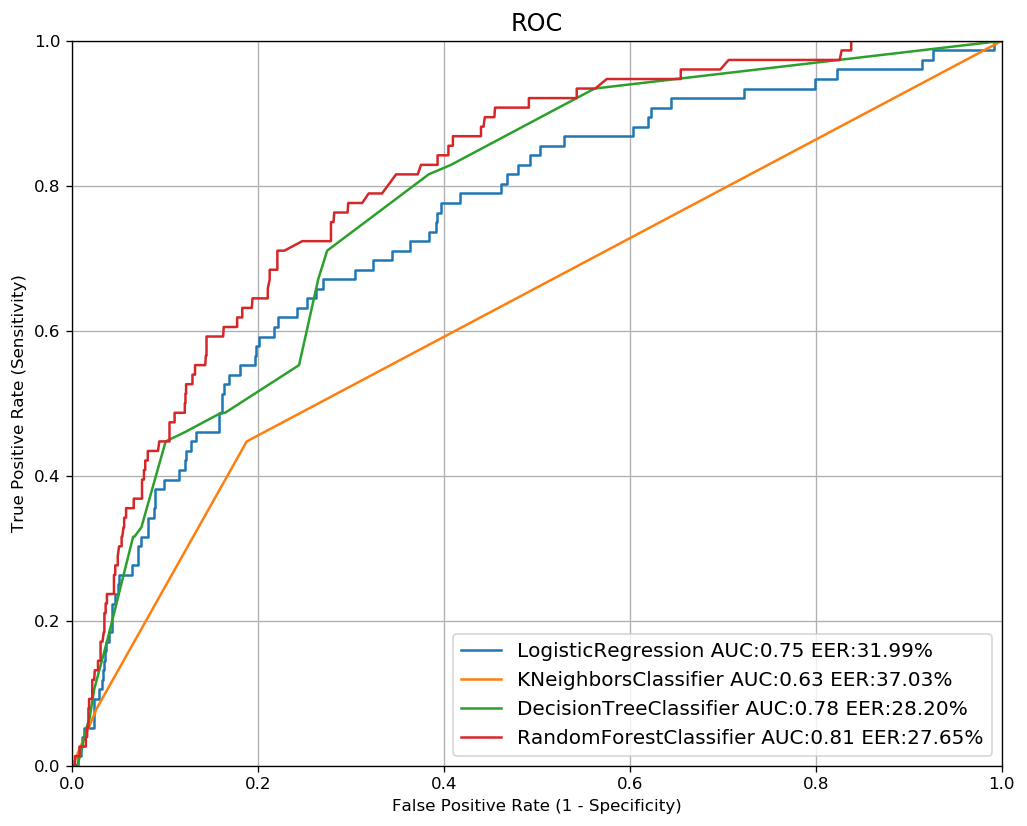
plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.rcParams['font.size'] = 12 plt.title('ROC')

plt.xlabel('False Positive Rate (1 - Specificity)') plt.ylabel('True Positive Rate (Sensitivity)')

plt.legend() plt.grid(**True**)



plt.figure(figsize=(10,8),dpi=120) classifiers = [

LogisticRegression(C=0.1,penalty='l2',random\_state=0), KNeighborsClassifier(5),

DecisionTreeClassifier(max\_depth=5),

RandomForestClassifier(max\_depth=5, n\_estimators=10)]

**for** model **in** classifiers:

model.fit(x\_train,y\_train)

y\_proba=model.predict\_proba(x\_test)[:,1]

plot\_roc(y\_test,y\_proba,type(model). name )

**from sklearn.preprocessing import** StandardScaler sscaler = StandardScaler()

x\_train = sscaler.fit\_transform(x\_train) x\_test = sscaler.transform(x\_test)

*# Standardization*

In [36]:

*# use randomforest as our model*

randomforest = RandomForestClassifier() randomforest.fit(x\_train, y\_train)

y\_pred = randomforest.predict(x\_test)

In [37]:

**from sklearn.metrics import** classification\_report, confusion\_matrix, accuracy\_score

print("The Classification report: **\n{}**".format(classification\_report(y\_test, y\_pred))) print(" ")

print("The Accuracy Score is: **{:.3f}**%".format(accuracy\_score(y\_test, y\_pred)\*100))

The Classification report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.95 | 1.00 | 0.97 | 1457 |
| 1 | 0.00 | 0.00 | 0.00 | 76 |
| accuracy |  |  | 0.95 | 1533 |
| macro avg | 0.48 | 0.50 | 0.49 | 1533 |
| weighted avg | 0.90 | 0.95 | 0.93 | 1533 |

The Accuracy Score is: 94.977%

In [38]:

importances = randomforest.feature\_importances\_ feat\_labels = x.columns[0:]

*# find the feature importances*

In [39]:

indices = np.argsort(importances)

**for** f **in** range(x\_train.shape[1]):

print("**%2d**) **%-\*s %f**" % (f + 1, 30, feat\_labels[indices[f]], importances[indices[f

]]))

*# show each featuers' importrances to stroke*

|  |  |
| --- | --- |
| 1) ever\_married | 0.020735 |
| 2) hypertension | 0.024168 |
| 3) heart\_disease | 0.029258 |
| 4) Residence\_type | 0.031888 |
| 5) gender | 0.034417 |
| 6) smoking\_status | 0.051633 |
| 7) work\_type | 0.053477 |
| 8) bmi | 0.234667 |
| 9) age | 0.240199 |
| 10) avg\_glucose\_level | 0.279557 |

In [41]:

threshold = 0.15

x\_selected = x\_train[:, importances > threshold] x\_selected.shape

*# age, bmi and avg\_glucose\_level are most significant value*

Out[41]:

(3577, 3)

In [42]:

plt.title('Feature Importance') plt.bar(range(x\_train.shape[1]),

importances[indices], align='center')

plt.xticks(range(x\_train.shape[1]),

feat\_labels[indices], rotation=90) plt.xlim([-1,x\_train.shape[1]])

plt.tight\_layout() plt.show()

