In [153…

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import export_graphviz
import graphviz


from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Reading the csv file of the dataset

In [154…

```
df = pd.read_csv("monica.csv")
```

In [155…

```
df.head()
```

Out[155…

	Serial No.	outcome	sex	age	yronset	premi	smstat	diabetes	highbp	hichol	angina	stroke	hosp
0	1	live	f	63	85	n	x	n	y	y	n	n	y
1	2	live	m	59	85	y	x	n	y	n	n	n	y
2	3	live	m	68	85	n	n	n	y	n	n	n	y
3	4	live	m	46	85	n	c	n	n	n	n	n	y
4	5	dead	m	48	85	n	n	y	n	n	y	n	y

In [156…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6367 entries, 0 to 6366
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Serial No.  6367 non-null   int64 
 1   outcome     6367 non-null   object
 2   sex         6367 non-null   object
 3   age         6367 non-null   int64 
 4   yronset     6367 non-null   int64 
 5   premi       6367 non-null   object
 6   smstat      6367 non-null   object
 7   diabetes    6367 non-null   object
 8   highbp      6367 non-null   object
 9   hichol      6367 non-null   object
 10  angina      6367 non-null   object
 11  stroke      6367 non-null   object
 12  hosp        6367 non-null   object
dtypes: int64(3), object(10)
memory usage: 646.8+ KB
```

In [157…
```
df.isnull().sum()
```

Out[157…
```
Serial No.    0
outcome       0
sex           0
age           0
yronset       0
premi         0
smstat        0
diabetes      0
highbp        0
hichol        0
angina        0
stroke        0
hosp          0
dtype: int64
```

In [158…
```
df.describe(include = 'all')
```

Out[158...

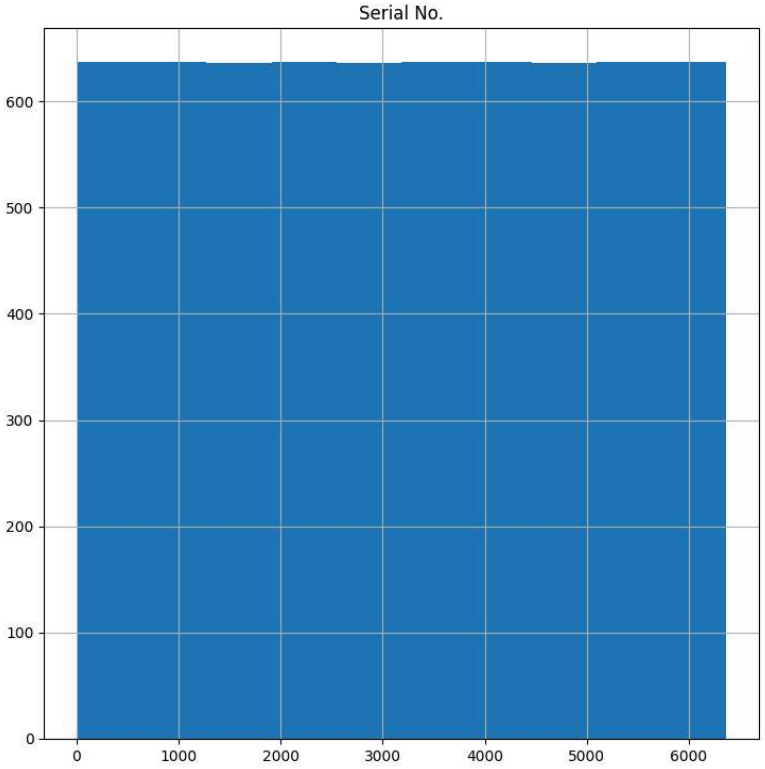
	Serial No.	outcome	sex	age	yronset	premi	smstat	diabetes	highbp	hichol	angina	stroke	hosp
count	6367.000000	6367	6367	6367.000000	6367.000000	6367	6367	6367	6367	6367	6367	6367	6367
unique	NaN	2	2	NaN	NaN	3	4	3	3	3	3	3	2
top	NaN	live	m	NaN	NaN	n	c	n	y	n	n	n	y
freq	NaN	3525	4605	NaN	NaN	4122	2051	4664	2877	3294	3473	4881	4442
mean	3184.000000	NaN	NaN	59.419978	88.749018	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	1838.138914	NaN	NaN	7.853923	2.558180	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	1.000000	NaN	NaN	35.000000	85.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	1592.500000	NaN	NaN	55.000000	87.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	3184.000000	NaN	NaN	61.000000	89.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	4775.500000	NaN	NaN	66.000000	91.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	6367.000000	NaN	NaN	69.000000	93.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

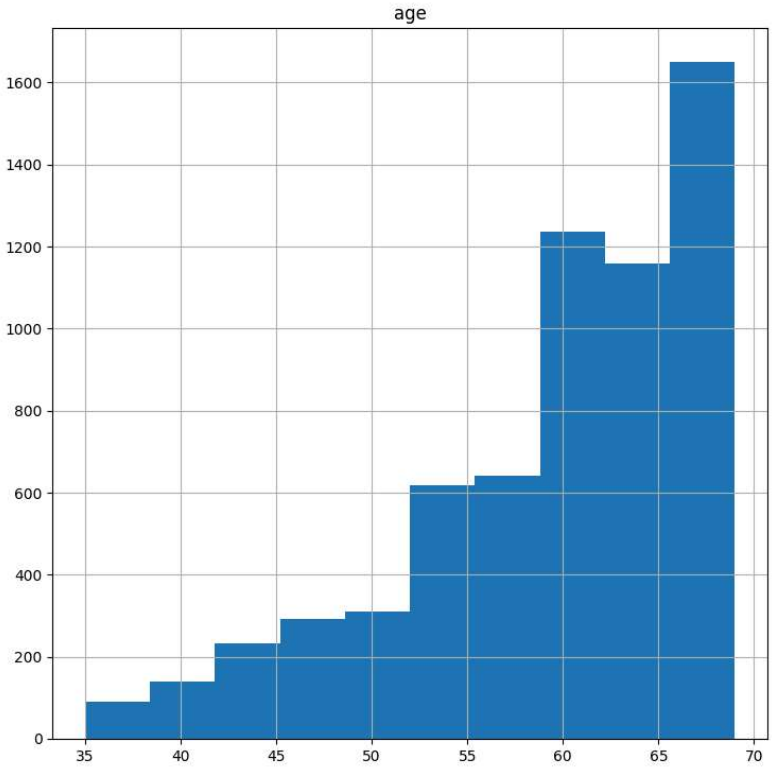
In [159...

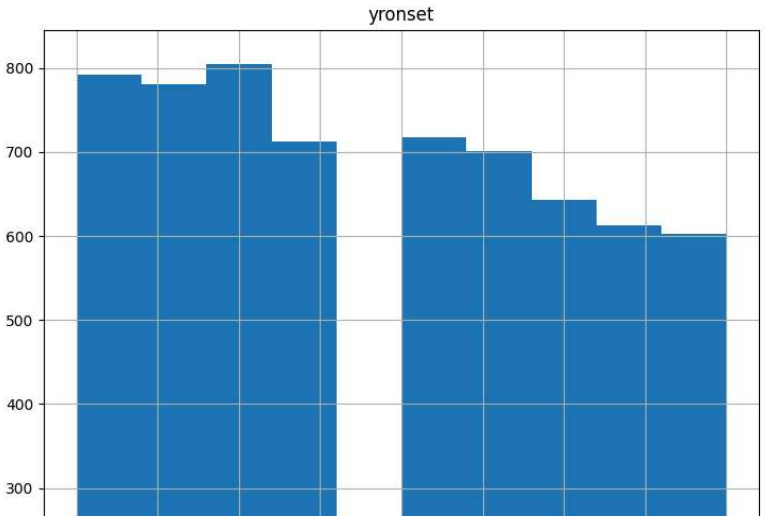
```
df.hist(figsize=(20,20)) #shows only numerical values
```

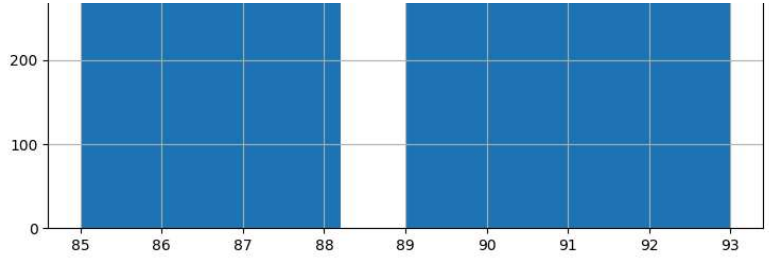
Out[159...

```
array([[<Axes: title={'center': 'Serial No.'}>,
        <Axes: title={'center': 'age'}>],
       [<Axes: title={'center': 'yronset'}>, <Axes: >]], dtype=object)
```


Serial No.
600
500
400
300
200
100
0
0
1000
2000
3000
4000
5000
6000


age
1600
1400
1200
1000
800
600
400
200
0
35
40
45
50
55
60
65
70


yronset
800
700
600
500
400
300



In [160…
```
df = df.drop(["Serial No."],axis=1)
```

Converting categorial data to numbers using LabelEncoder(). But by default our dataset in of 'object' type for LabelEncoder to work we'll have to convert to 'catergory' type.

In [161…
```
df = df.astype('category')
```

In [162…
```
df.dtypes # or use df.info()
```

Out[162…
```
outcome     category
sex         category
age         category
yronset     category
premi       category
smstat      category
diabetes    category
highbp      category
hichol      category
angina      category
stroke      category
hosp        category
dtype: object
```

LabelEncoder

In [163…
```
labelencoder=LabelEncoder()
for column in df.columns:
    df[column] = labelencoder.fit_transform(df[column])
```

In [164…
```
df.head()
```

Out[164…

	outcome	sex	age	yronset	premi	smstat	diabetes	highbp	hichol	angina	stroke	hosp
0	1	0	28	0	0	3	0	2	2	0	0	1
1	1	1	24	0	2	3	0	2	0	0	0	1
2	1	1	33	0	0	1	0	2	0	0	0	1
3	1	1	11	0	0	0	0	0	0	0	0	1
4	0	1	13	0	0	1	2	0	0	2	0	1

In [165…
```
df['yronset'].unique()
```

Out[165…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8], dtype=int64)
```

In [166…
```
df["yronset"].value_counts()
```

Out[166…
```
yronset
2    805
0    792
1    781
4    718
3    712
5    701
6    643
7    612
8    603
Name: count, dtype: int64
```

In [167…
```
df["stroke"].value_counts()
```

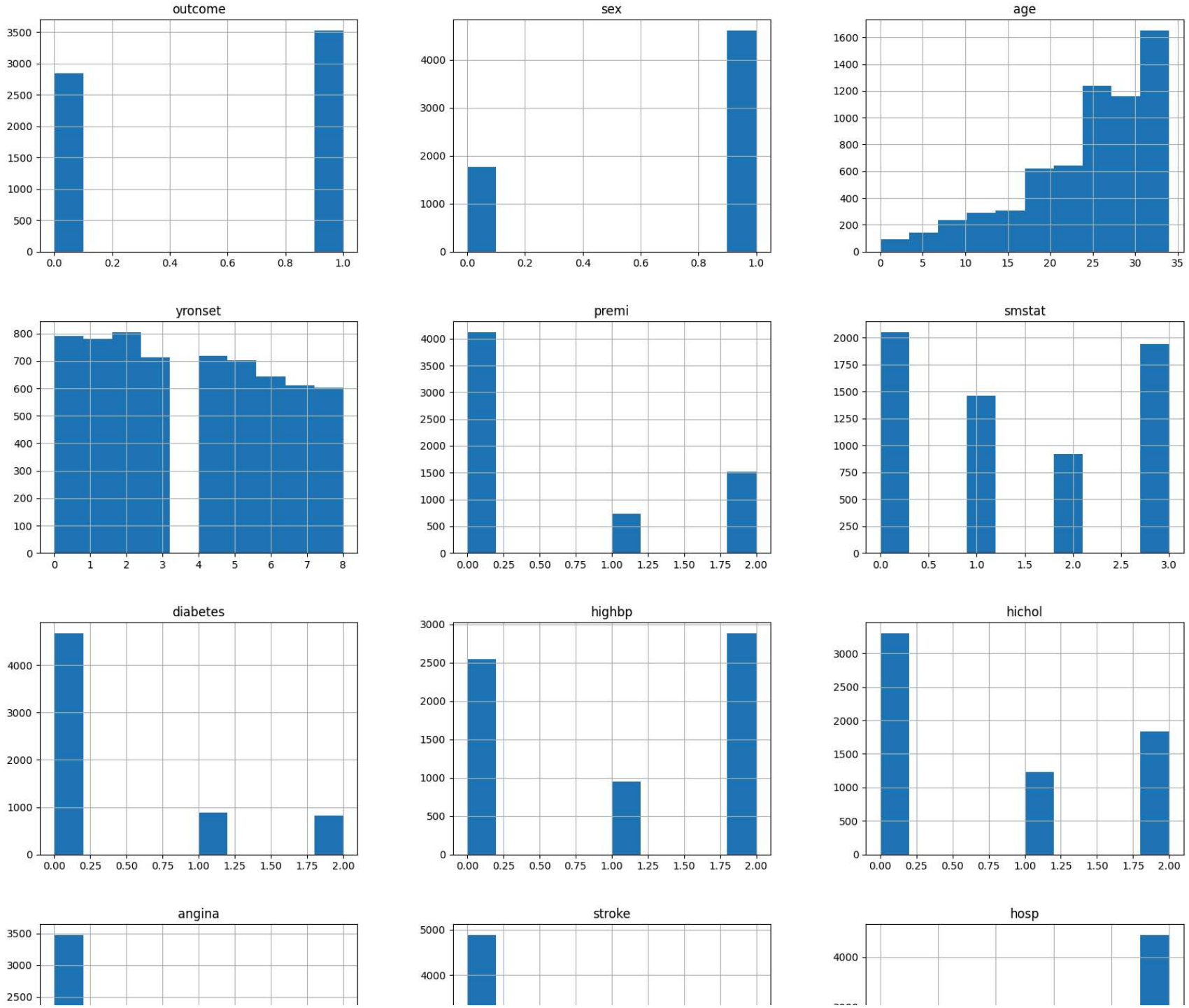
Out[167…
```
stroke
0    4881
1     926
2     560
Name: count, dtype: int64
```

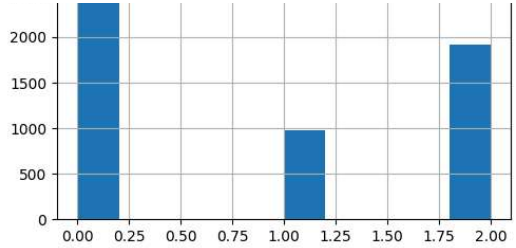
In [168…
```
df["hosp"].value_counts()
```

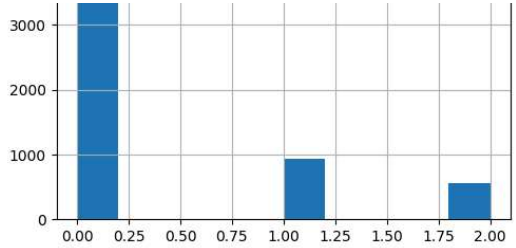
Out[168…
```
hosp
1    4442
0    1925
Name: count, dtype: int64
```

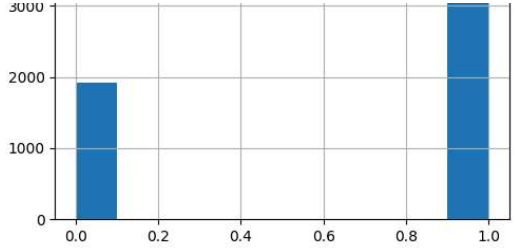
In [169…
```
df.hist(figsize=(20,20))
```

Out[169…
```
array([[<Axes: title={'center': 'outcome'}>,
        <Axes: title={'center': 'sex'}>, <Axes: title={'center': 'age'}>],
       [<Axes: title={'center': 'yronset'}>,
        <Axes: title={'center': 'premi'}>,
        <Axes: title={'center': 'smstat'}>],
       [<Axes: title={'center': 'diabetes'}>,
        <Axes: title={'center': 'highbp'}>,
        <Axes: title={'center': 'hichol'}>],
       [<Axes: title={'center': 'angina'}>,
        <Axes: title={'center': 'stroke'}>,
        <Axes: title={'center': 'hosp'}>]], dtype=object)
```


outcome
sex
age
yronset
premi
smstat
diabetes
highbp
hichol
angina
stroke
hosp





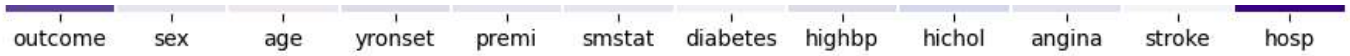


Corelation

In [170…

```
plt.figure(figsize=(14,12))
sns.heatmap(df.corr(),linewidths=.1,cmap="Purples", annot=True, annot_kws={"size": 7})
plt.yticks(rotation=0);
```


outcome 1 0.00036 -0.16 0.07 -0.1 -0.11 -0.19 -0.006 0.033 -0.11 -0.27 0.73
sex 0.00036 1 -0.12 0.0072 0.041 0.087 -0.088 -0.14 -0.024 -0.024 -0.037 -0.05
age -0.16 -0.12 1 -0.0055 0.089 0.18 0.092 0.11 -0.027 0.18 0.14 -0.076
yronset 0.07 0.0072 -0.0055 1 0.0053 0.046 0.029 0.0068 0.15 -0.021 -0.019 0.032
premi -0.1 0.041 0.089 0.0053 1 0.19 0.11 0.019 0.14 0.43 0.12 -0.042
smstat -0.11 0.087 0.18 0.046 0.19 1 0.1 0.073 0.08 0.16 0.098 -0.068
diabetes -0.19 -0.088 0.092 0.029 0.11 0.1 1 0.081 0.057 0.14 0.24 -0.14
highbp -0.006 -0.14 0.11 0.0068 0.019 0.073 0.081 1 0.15 0.097 0.11 0.021
hichol 0.033 -0.024 -0.027 0.15 0.14 0.08 0.057 0.15 1 0.17 0.091 0.047
angina -0.11 -0.024 0.18 -0.021 0.43 0.16 0.14 0.097 0.17 1 0.17 -0.023
stroke -0.27 -0.037 0.14 -0.019 0.12 0.098 0.24 0.11 0.091 0.17 1 -0.19
hosp 0.73 -0.05 -0.076 0.032 -0.042 -0.068 -0.14 0.021 0.047 -0.023 -0.19 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



In [171…
```
# df = df.drop(["stroke","angina","diabetes","smstat"],axis=1)
```

Split Dataset

In [172…
```
X = df.drop(['outcome'], axis=1)
y = df["outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)
```

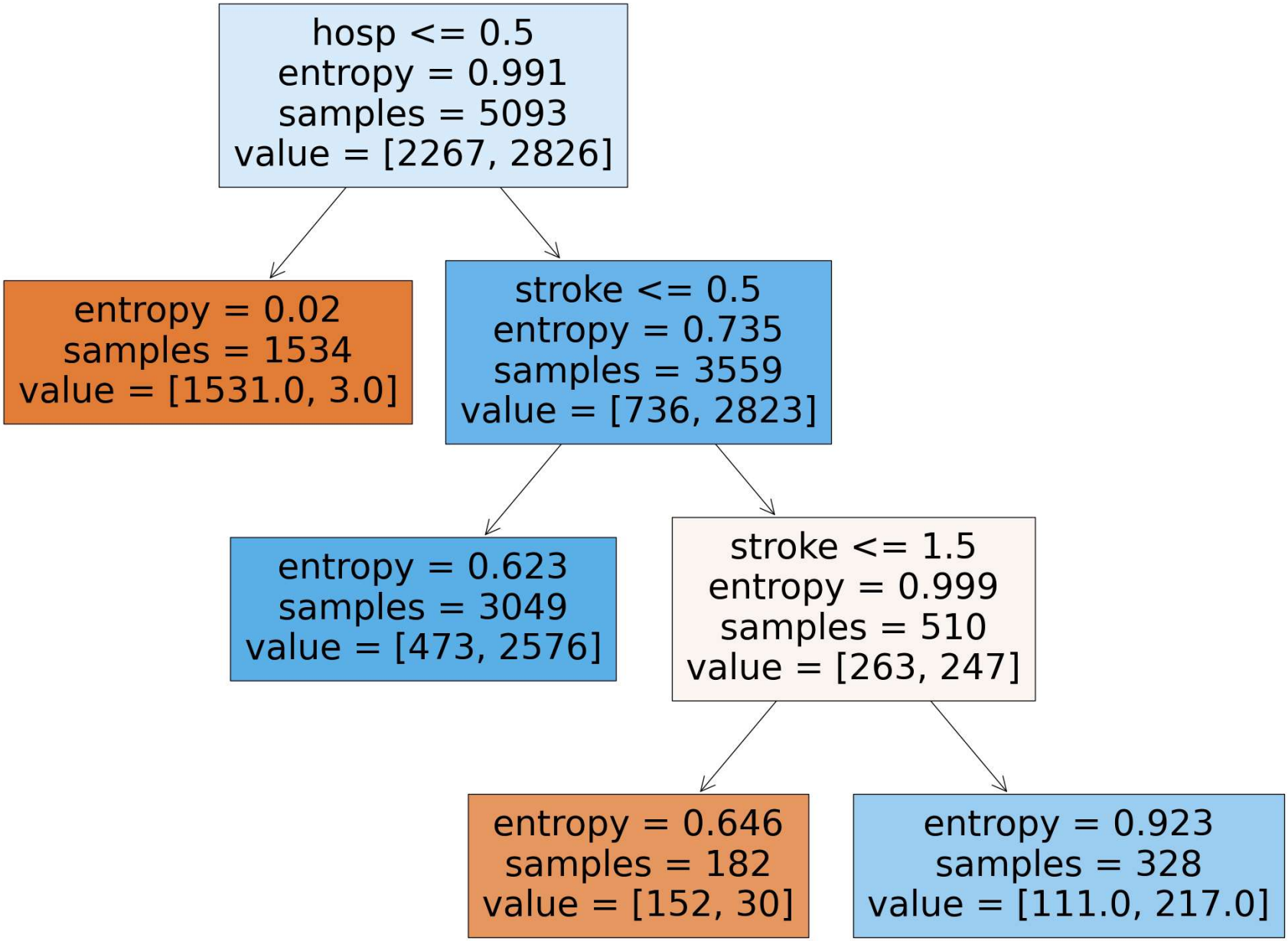
In [173…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

dt = DecisionTreeClassifier(criterion = 'entropy', random_state = 0,ccp_alpha=0.015)
dt.fit(X_train, y_train)
```

Out[173…
DecisionTreeClassifier

```
DecisionTreeClassifier(ccp_alpha=0.015, criterion='entropy', random_state=0)
```

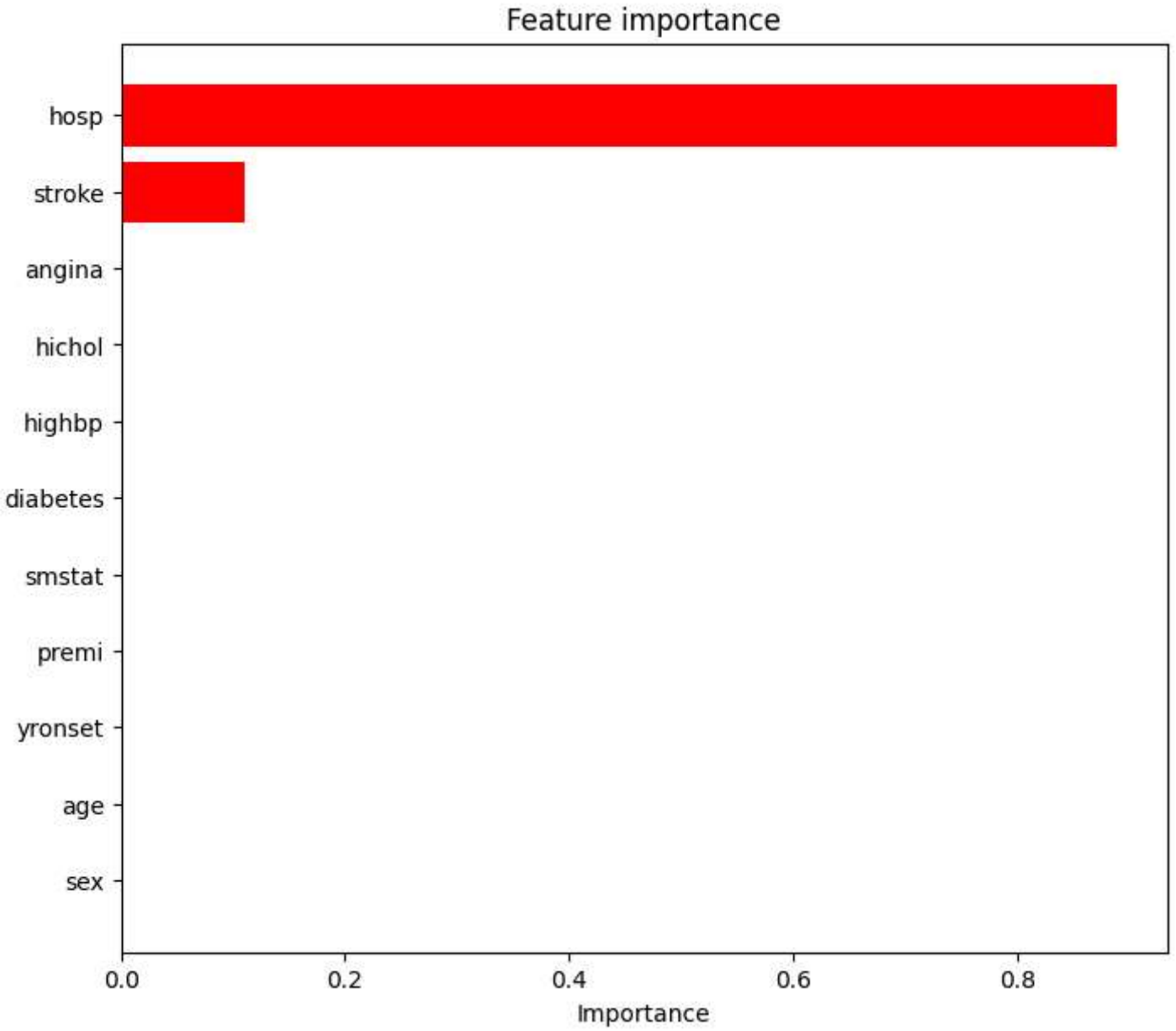
In [174…
```
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(dt,
                   feature_names=X.columns,
                   filled=True)
```


hosp <= 0.5
entropy = 0.991
samples = 5093
value = [2267, 2826]
entropy = 0.02
samples = 1534
value = [1531.0, 3.0]
stroke <= 0.5
entropy = 0.735
samples = 3559
value = [736, 2823]
entropy = 0.623
samples = 3049
value = [473, 2576]
stroke <= 1.5
entropy = 0.999
samples = 510
value = [263, 247]
entropy = 0.646
samples = 182
value = [152, 30]
entropy = 0.923
samples = 328
value = [111.0, 217.0]

In [175…

```
features_list = X.columns.values
feature_importance = dt.feature_importances_
sorted_idx = np.argsort(feature_importance)

plt.figure(figsize=(8,7))
plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx], align='center', color ="red")
plt.yticks(range(len(sorted_idx)), features_list[sorted_idx])
plt.xlabel('Importance')
plt.title('Feature importance')
plt.draw()
plt.show()
```



Model Validation

In [176…
```
y_pred_dt = dt.predict(X_test)
```

In [177…
```
print("Decision Tree Classifier report: \n\n", classification_report(y_test, y_pred_dt))
```

```
Decision Tree Classifier report:

                precision    recall  f1-score   support

           0       0.97      0.74      0.84       575
           1       0.82      0.98      0.90       699

    accuracy                           0.88      1274
   macro avg       0.90      0.86      0.87      1274
weighted avg       0.89      0.88      0.87      1274
```

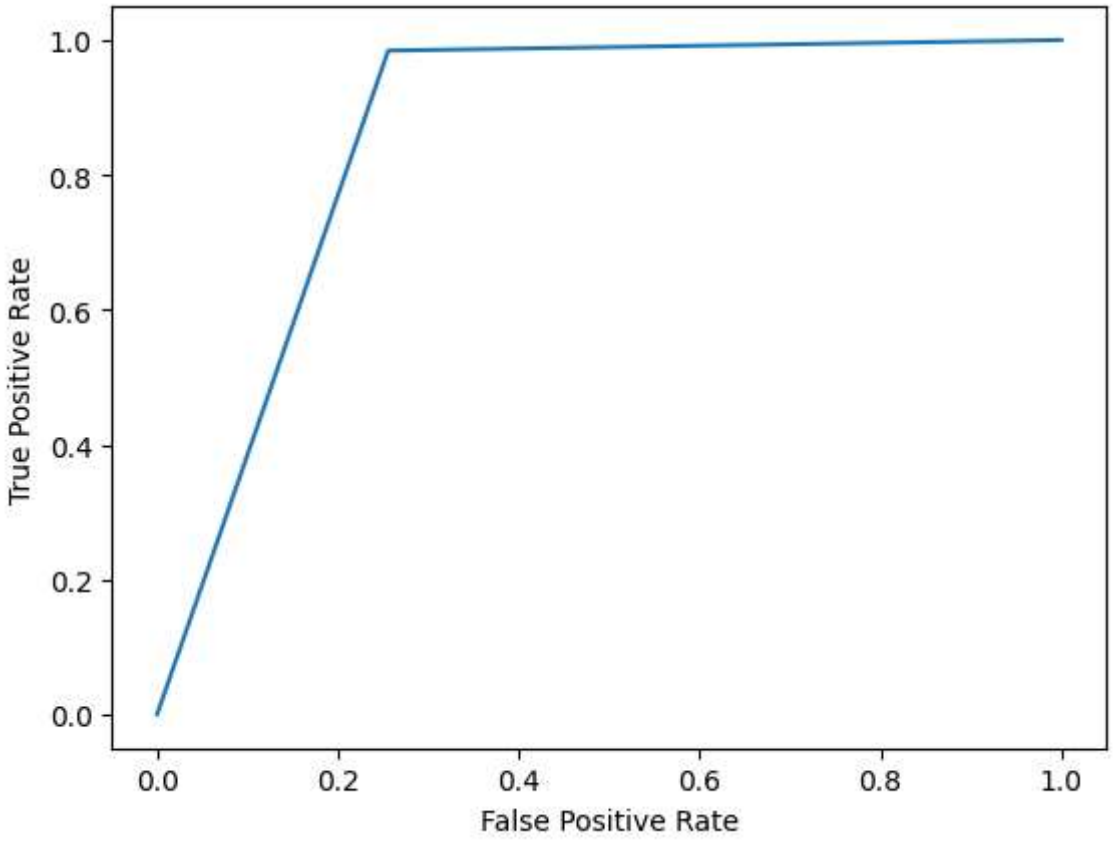
In [178…
```
print("Test Accuracy: {}%".format(round(dt.score(X_test, y_test)*100, 2)))
```

```
Test Accuracy: 87.6%
```

ROC curve

In [179…
```
from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, threshold = roc_curve(y_test, y_pred_dt)
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
roc_auc_score(y_test, y_pred_dt)
```

Out[179…
```
0.8643055296386142
```



Logistic Regression

Feature Scaling

In [180…

```
sc = StandardScaler()
X = sc.fit_transform(X)
X
```

Out[180…

```
array([[-1.61663458,  0.45586178, -1.46561704, ..., -0.85194584,
        -0.51195126,  0.65830339],
       [ 0.61856898, -0.05347786, -1.46561704, ..., -0.85194584,
        -0.51195126,  0.65830339],
       [ 0.61856898,  1.09253634, -1.46561704, ..., -0.85194584,
        -0.51195126,  0.65830339],
       ...,
       [ 0.61856898,  0.20119196,  1.66185132, ..., -0.85194584,
        -0.51195126,  0.65830339],
       [-1.61663458,  0.71053161,  1.66185132, ..., -0.85194584,
        -0.51195126,  0.65830339],
       [ 0.61856898, -1.96350154,  1.66185132, ..., -0.85194584,
        -0.51195126,  0.65830339]])
```

Splitting dataset

In [181…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4456, 11)
(1911, 11)
(4456,)
(1911,)
```

Fitting the logistic regression model and predicting test results

In [182…

```
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

Out[182...

```
LogisticRegression()
```

In [183...

```
y_pred = classifier.predict(X_test)
```

In [184...

```
result = pd.DataFrame({'Actual' : y_test, 'Predicted' : y_pred})
result
```

Out[184...

	Actual	Predicted
3432	0	0
5345	0	0
1686	0	1
1576	1	1
4680	1	1
...	...	...
4417	0	0
3816	1	1
1094	1	1
2167	0	0
5739	0	0

1911 rows × 2 columns

Coefficient and Intercept

In [185...

```
classifier.coef_
```

Out[185…
```
array([[ 0.12794104, -0.32162399,  0.18887339, -0.1284792 , -0.16842471,
        -0.22027442,  0.08888107,  0.00843566, -0.1530953 , -0.35388724,
         3.52996258]])
```

In [186…
```
classifier.intercept_
```

Out[186…
```
array([-0.93225359])
```

Evaluating the model

In [187…
```
print(classifier.predict_proba(X))
```

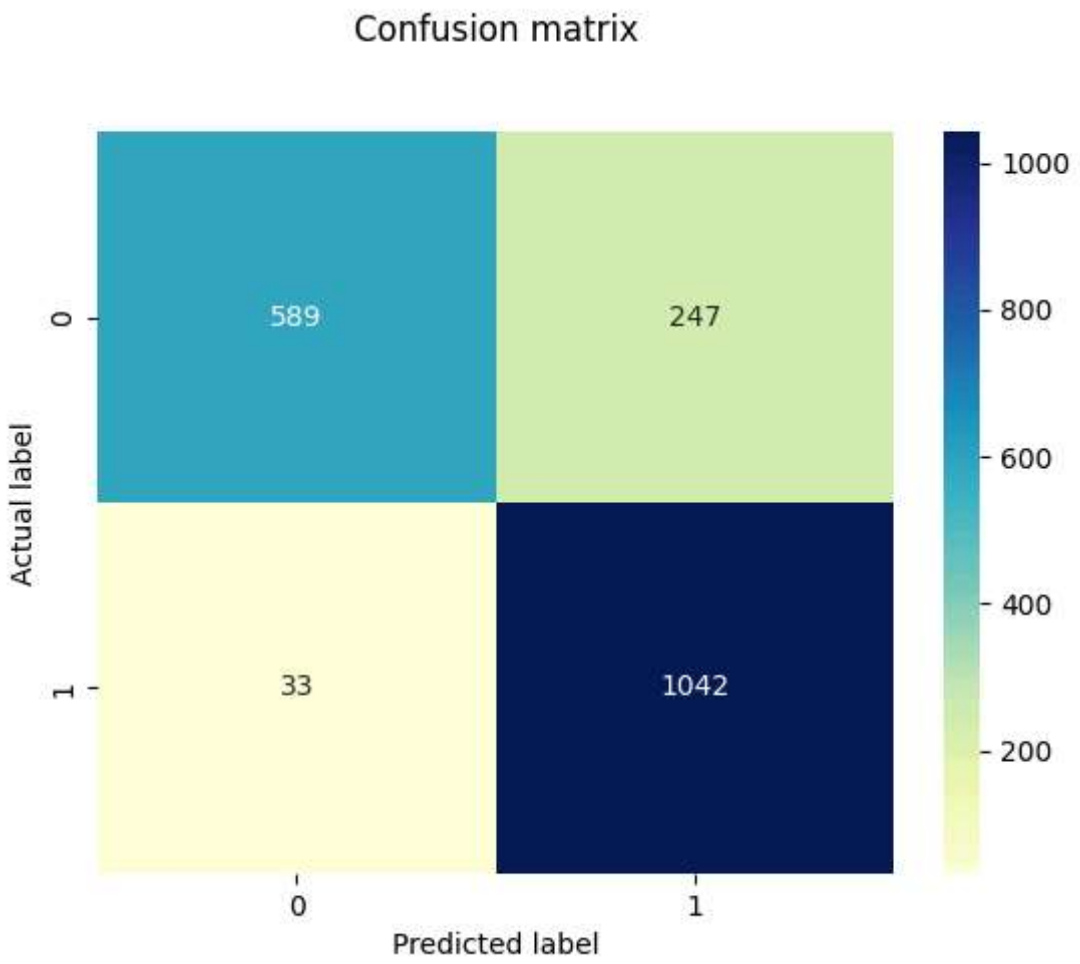
```
[[0.2361931  0.7638069 ]
 [0.21408563 0.78591437]
 [0.18084383 0.81915617]
 ...
 [0.09846103 0.90153897]
 [0.14742536 0.85257464]
 [0.03835241 0.96164759]]
```

In [188…
```
cf_matrix = confusion_matrix(y_test, y_pred)
cf_matrix
```

Out[188…
```
array([[ 589,  247],
       [  33, 1042]], dtype=int64)
```

In [189…
```
sns.heatmap(pd.DataFrame(cf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[189…
```
Text(0.5, 23.52222222222222, 'Predicted label')
```



In [190…
```
accuracy_score(y_test,y_pred)
```

Out[190…
```
0.8534798534798534
```

In [191…
```
target_names = ['Fail', 'Pass']
print(classification_report(y_test, y_pred,target_names=target_names))
```

	precision	recall	f1-score	support
Fail	0.95	0.70	0.81	836
Pass	0.81	0.97	0.88	1075
accuracy			0.85	1911
macro avg	0.88	0.84	0.84	1911
weighted avg	0.87	0.85	0.85	1911

In [ ]: