Risk Factors for Hospitalization and Death from COVID-19 in Humanitarian Settings

Introduction to Data Mining Final Project Presentation

Gizachew Demessie

June 20222

Risk Factors for Hospitalization and Death from COVID-19 in Humanitarian Settings

- Dataset source: Google data search
  - Risk Factors for Hospitalization and Death from COVID-19 in Humanitarian Settings - Humanitarian Data Exchange (humdata.org)
- Data set provided by: Johns Hopkins Bloomberg School of Public Health
  - csv(275777)
  - Dataset updated

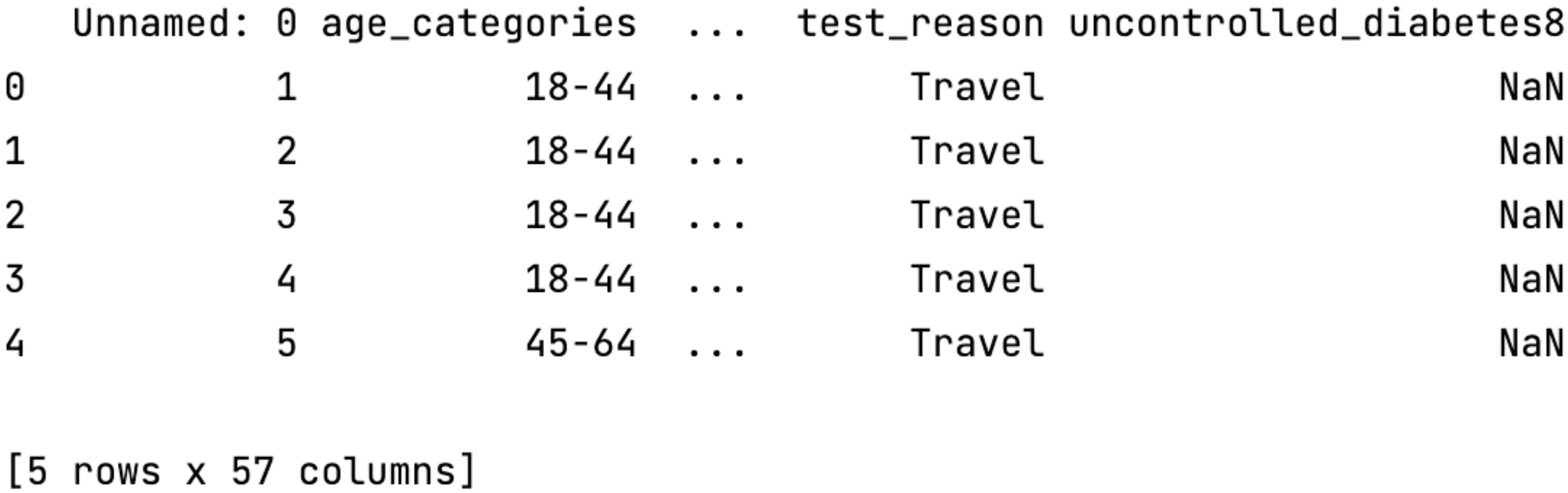
    Apr 12, 2022
- Data description
  - Deidentified dataset used for analysis presented in *"Risk Factors for Hospitalization and Death from COVID-19: A Prospective Cohort Study in South Sudan and Eastern Democratic Republic of the Congo"* by Leidman et al.

Data processing tools

- Python packages, and
- sklearn

Data preprocessing

- Get the data
  - pd.read_csv(‘covid_risk_factors.csv’)
- look at first few rows
  - print(df.head()



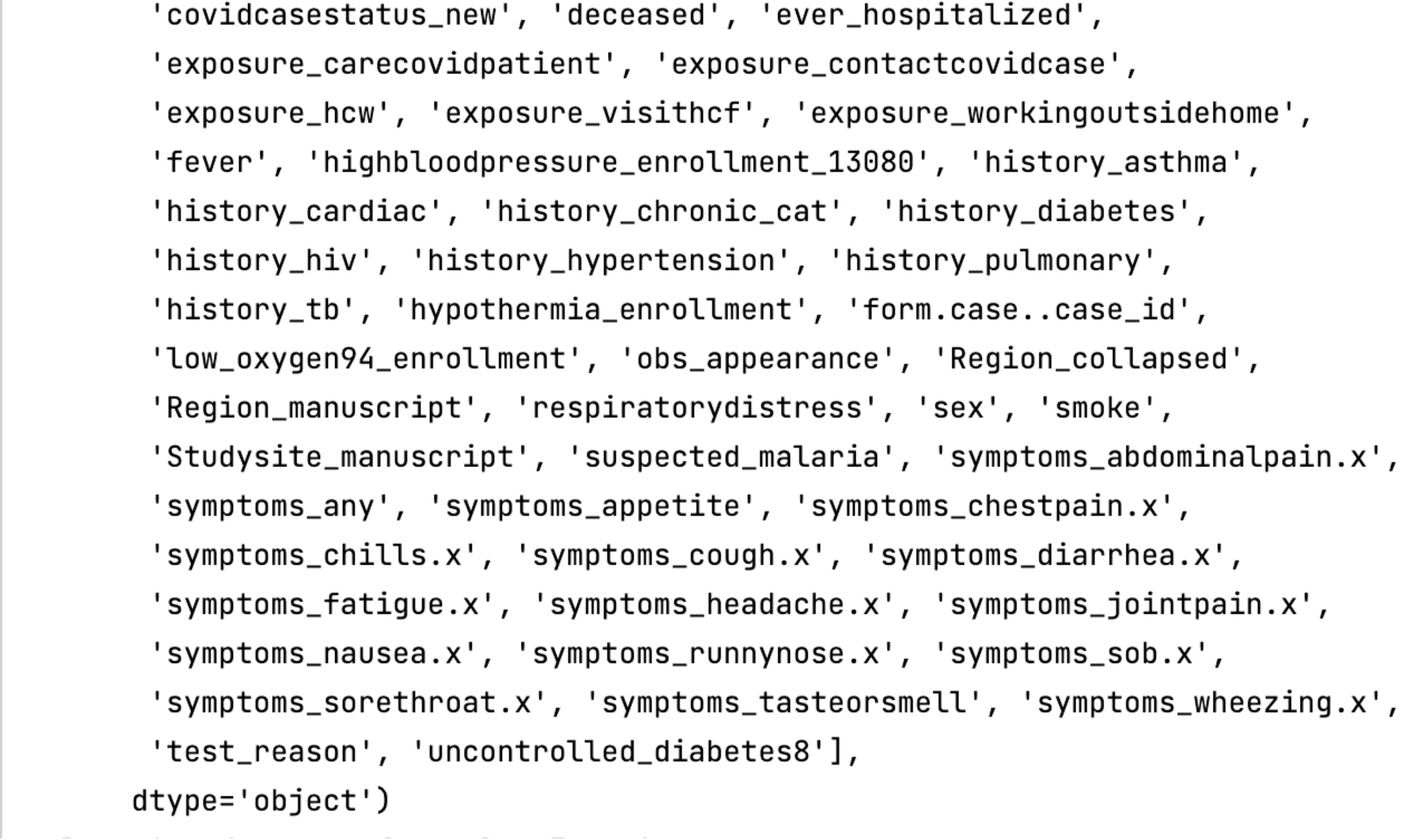
```
   Unnamed: 0 age_categories  ...  test_reason uncontrolled_diabetes8
0           1          18-44  ...       Travel                    NaN
1           2          18-44  ...       Travel                    NaN
2           3          18-44  ...       Travel                    NaN
3           4          18-44  ...       Travel                    NaN
4           5          45-64  ...       Travel                    NaN

[5 rows x 57 columns]
```

- (Data frame shape: 519, 57)

EDA

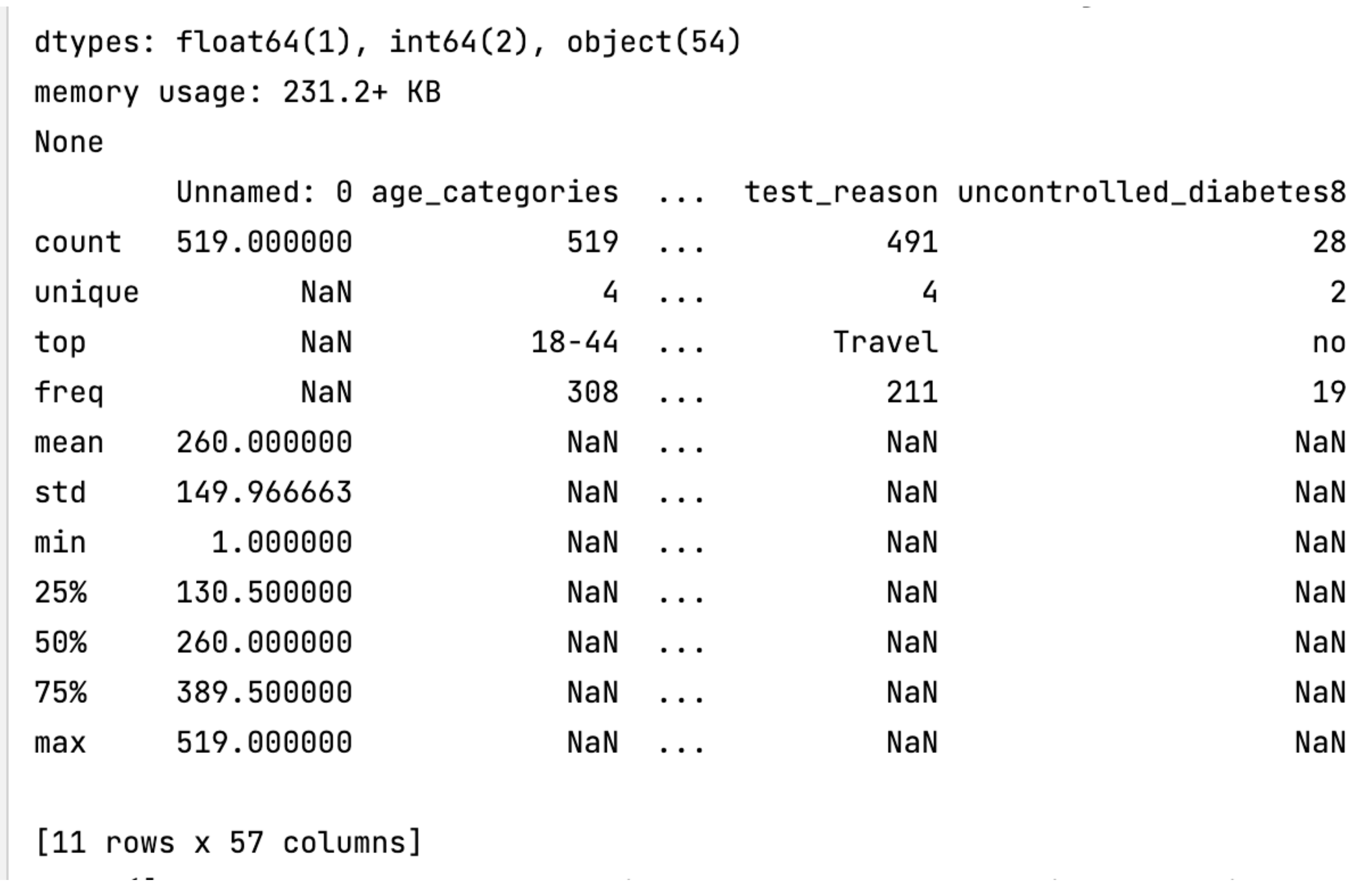
- Columns



```
       'covidcasestatus_new', 'deceased', 'ever_hospitalized',
       'exposure_carecovidpatient', 'exposure_contactcovidcase',
       'exposure_hcw', 'exposure_visithcf', 'exposure_workingoutsidehome',
       'fever', 'highbloodpressure_enrollment_13080', 'history_asthma',
       'history_cardiac', 'history_chronic_cat', 'history_diabetes',
       'history_hiv', 'history_hypertension', 'history_pulmonary',
       'history_tb', 'hypothermia_enrollment', 'form.case..case_id',
       'low_oxygen94_enrollment', 'obs_appearance', 'Region_collapsed',
       'Region_manuscript', 'respiratorydistress', 'sex', 'smoke',
       'Studysite_manuscript', 'suspected_malaria', 'symptoms_abdominalpain.x',
       'symptoms_any', 'symptoms_appetite', 'symptoms_chestpain.x',
       'symptoms_chills.x', 'symptoms_cough.x', 'symptoms_diarrhea.x',
       'symptoms_fatigue.x', 'symptoms_headache.x', 'symptoms_jointpain.x',
       'symptoms_nausea.x', 'symptoms_runnynose.x', 'symptoms_sob.x',
       'symptoms_sorethroat.x', 'symptoms_tasteorsmell', 'symptoms_wheezing.x',
       'test_reason', 'uncontrolled_diabetes8'],
      dtype='object')
```

EDA

- check the structure of data `print(df.info())`
- check the summary of the data `print(df.describe(include='all’)`



```
dtypes: float64(1), int64(2), object(54)
memory usage: 231.2+ KB
None
        Unnamed: 0 age_categories  ...  test_reason uncontrolled_diabetes8
count   519.000000            519  ...          491                     28
unique         NaN              4  ...            4                      2
top            NaN          18-44  ...       Travel                     no
freq           NaN            308  ...          211                     19
mean    260.000000            NaN  ...          NaN                    NaN
std     149.966663            NaN  ...          NaN                    NaN
min       1.000000            NaN  ...          NaN                    NaN
25%     130.500000            NaN  ...          NaN                    NaN
50%     260.000000            NaN  ...          NaN                    NaN
75%     389.500000            NaN  ...          NaN                    NaN
max     519.000000            NaN  ...          NaN                    NaN

[11 rows x 57 columns]
```

EDA

- Select important features that have potential effect on the outcome (risk factors that potentially affect the outcome of covid infection)
- Select columns by index;

```
df1 = df.iloc[: ,[1, 5, 7, 18,19, 22, 30, 35, 36, 37, 40 , 55, 12, 11]].copy()
print(df.columns)
```

-
  ```
  Index(['age_categories', 'anyinfectious', 'bmi_cat', 'fever',
        #'highbloodpressure_enrollment_13080', 'history_chronic_cat',
        #'low_oxygen94_enrollment', 'sex', 'smoke', 'Studysite_manuscript',
        #'symptoms_any', 'test_reason', 'ever_hospitalized', 'deceased'],dtype='object’)
  ```
- Df1 shape = (519, 11)

Data cleaning

- Identify unique values of each column

```
        print(df1[‘column_name'].unique())
```

- Change longer column names to shorter (easier to use)

```
        df1.rename(columns= {'highbloodpressure_enrollment_13080':'highbloodpressure',
                             'low_oxygen94_enrollment': 'low oxygen',
                             'ever_hospitalized': 'hospitalized'}, inplace=True)
```

- check the null values in each column

```
        print(df1.isnull().sum())
```

- drop columns with significant number of Nan

```
        df1=df1.drop('anyinfectious’), axis=1). # ‘anyinfectious’ has  #223/519 null
```

- replace categorical data with the most frequent value in that column

```
        df1 = df1.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

- check the null values again in each column

```
        print(df1.isnull().sum())
```

Data cleaning

- Replace string values with numerical values;
  - Create a dictionary of categorical values and numeric values to replace the categorical values by number.

```
age_cat=df1['age_categories'].unique()
age_num= np.arange(4)
dict_age=dict(zip(age_cat, age_num))
df1['age_categories'] = df1['age_categories'].replace(dict_age)
```

- Feature selection

```
from sklearn.feature_selection import VarianceThreshold
df1 = df1.drop('hospitalized', axis=1)
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
sel.fit_transform(dfX)
print(df1.head)
```

- No feature dropped by using the variance threshold feature selection

Use the Data mining models

- *split the dataset*

```
      X= df1.drop(['hospitalized'],axis=1)
      y= df1['hospitalized’]
```

- *Splitting the dataset into train and test*

```
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=100)
```

- *Standardize the Variables*

```
      stdsc = StandardScaler()
      stdsc.fit(X_train)
      X_train = stdsc.transform(X_train)
      X_test= stdsc.transform(X_test)
```

- *Train the model*
- *perform prediction*
- *Metrix Evaluations: Confusion matrix, accuracy*

Metrix evaluation of different models

Logit reg		*True value*	
		0	1
Pred value	*0*	109	2
	1	27	18

Accuracy = 0.81

KNN		*True value*	
		0	1
Pred value	*0*	97	14
	1	26	19

Accuracy = 0.74

SVM		*True value*	
		0	1
Pred value	*0*	108	3
	1	26	19

Accuracy = 0.81

DT		*True value*	
		0	1
Pred value	*0*	93	18
	1	24	21

Accuracy = 0.73

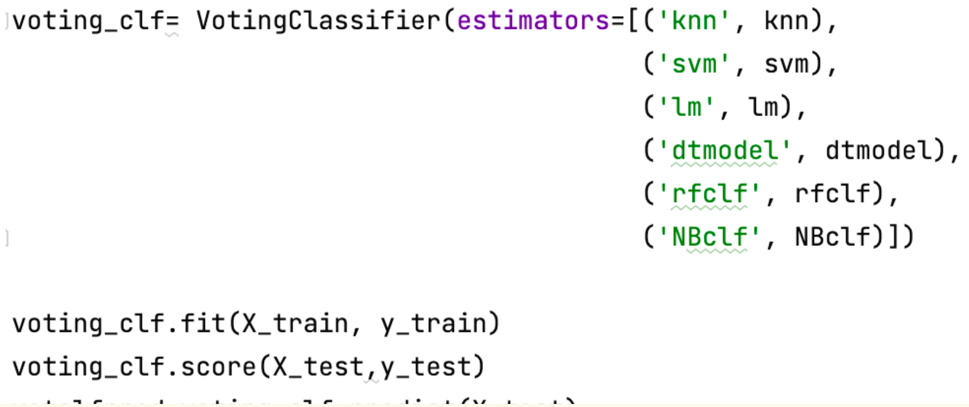
RF		*True value*	
		0	1
Pred value	*0*	99	12
	1	22	23

Accuracy = 0.78

NB		*True value*	
		0	1
Pred value	*0*	99	12
	1	22	23

Accuracy = 0.81

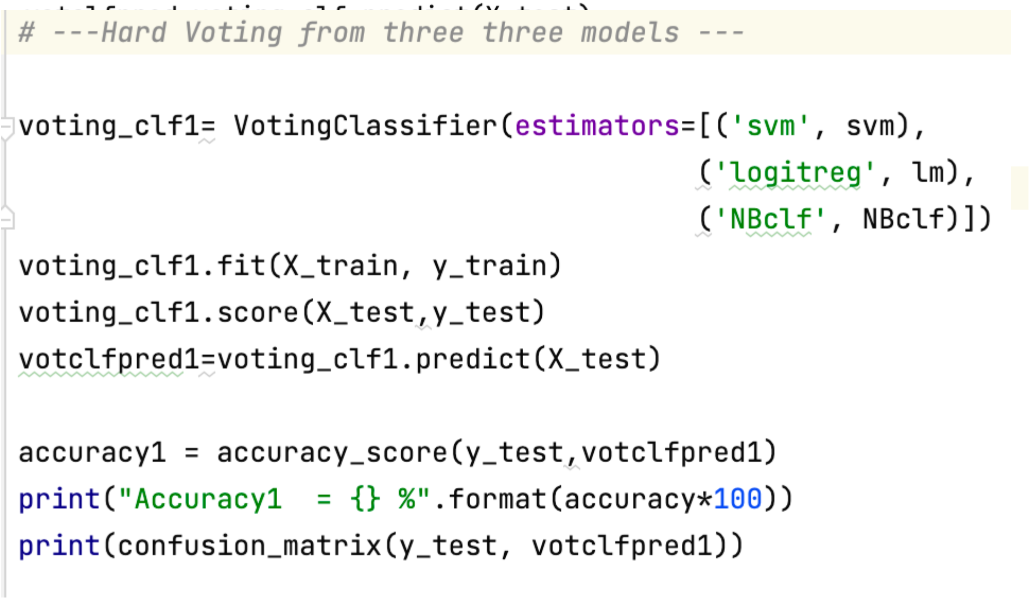
Apply voting classifier



```
voting_clf= VotingClassifier(estimators=[('knn', knn),
                                         ('svm', svm),
                                         ('lm', lm),
                                         ('dtmodel', dtmodel),
                                         ('rfclf', rfclf),
                                         ('NBclf', NBclf)])

voting_clf.fit(X_train, y_train)
voting_clf.score(X_test,y_test)
votclfpred=voting_clf.predict(X_test)
```

Accuracy = 0.79



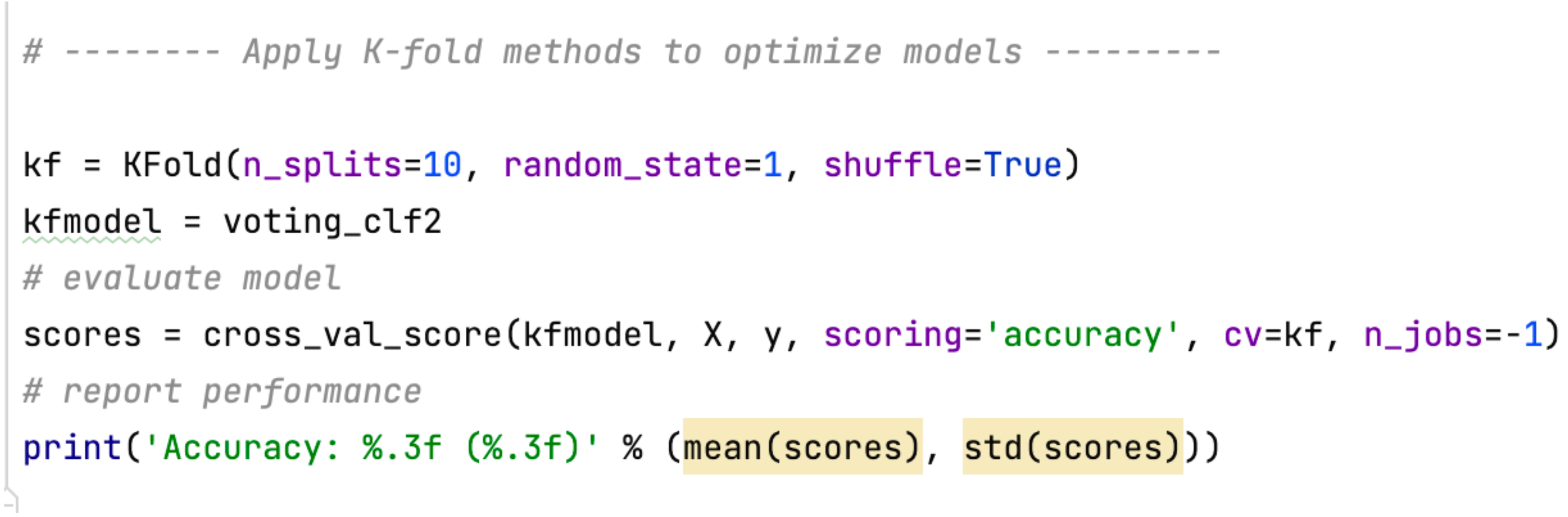
```
# ---Hard Voting from three three models ---

voting_clf1= VotingClassifier(estimators=[('svm', svm),
                                          ('logitreg', lm),
                                          ('NBclf', NBclf)])
voting_clf1.fit(X_train, y_train)
voting_clf1.score(X_test,y_test)
votclfpred1=voting_clf1.predict(X_test)

accuracy1 = accuracy_score(y_test,votclfpred1)
print("Accuracy1  = {} %".format(accuracy*100))
print(confusion_matrix(y_test, votclfpred1))
```

Accuracy 1= 0.78

Apply K-fold cross validation

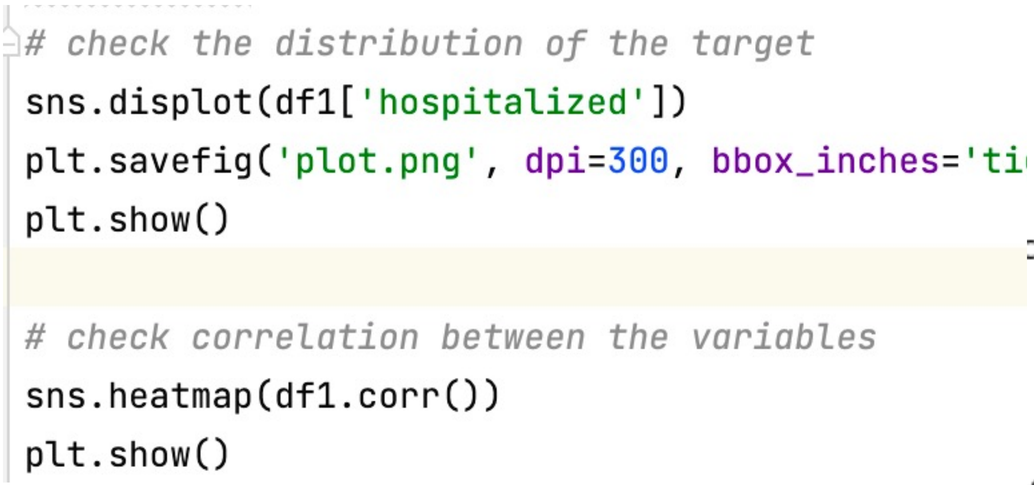


```
# -------- Apply K-fold methods to optimize models ---------

kf = KFold(n_splits=10, random_state=1, shuffle=True)
kfmodel = voting_clf2
# evaluate model
scores = cross_val_score(kfmodel, X, y, scoring='accuracy', cv=kf, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

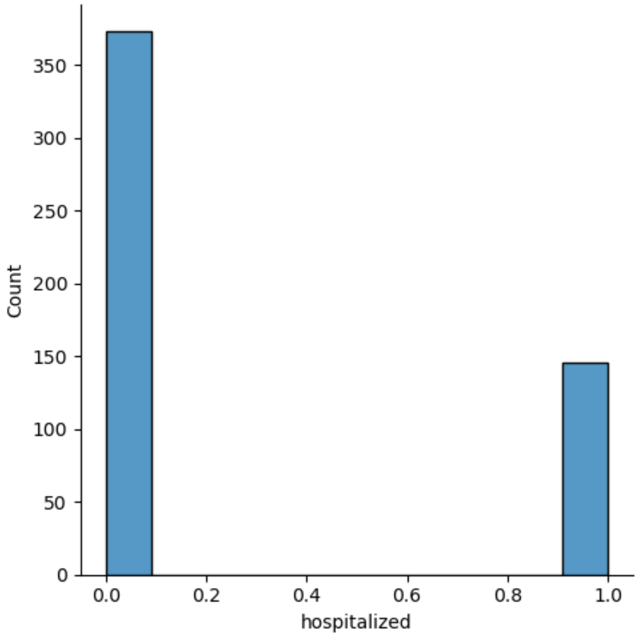
K=10, Accuracy: 0.78 (0.056)
K=5, Accuracy: 0.777 (0.055)

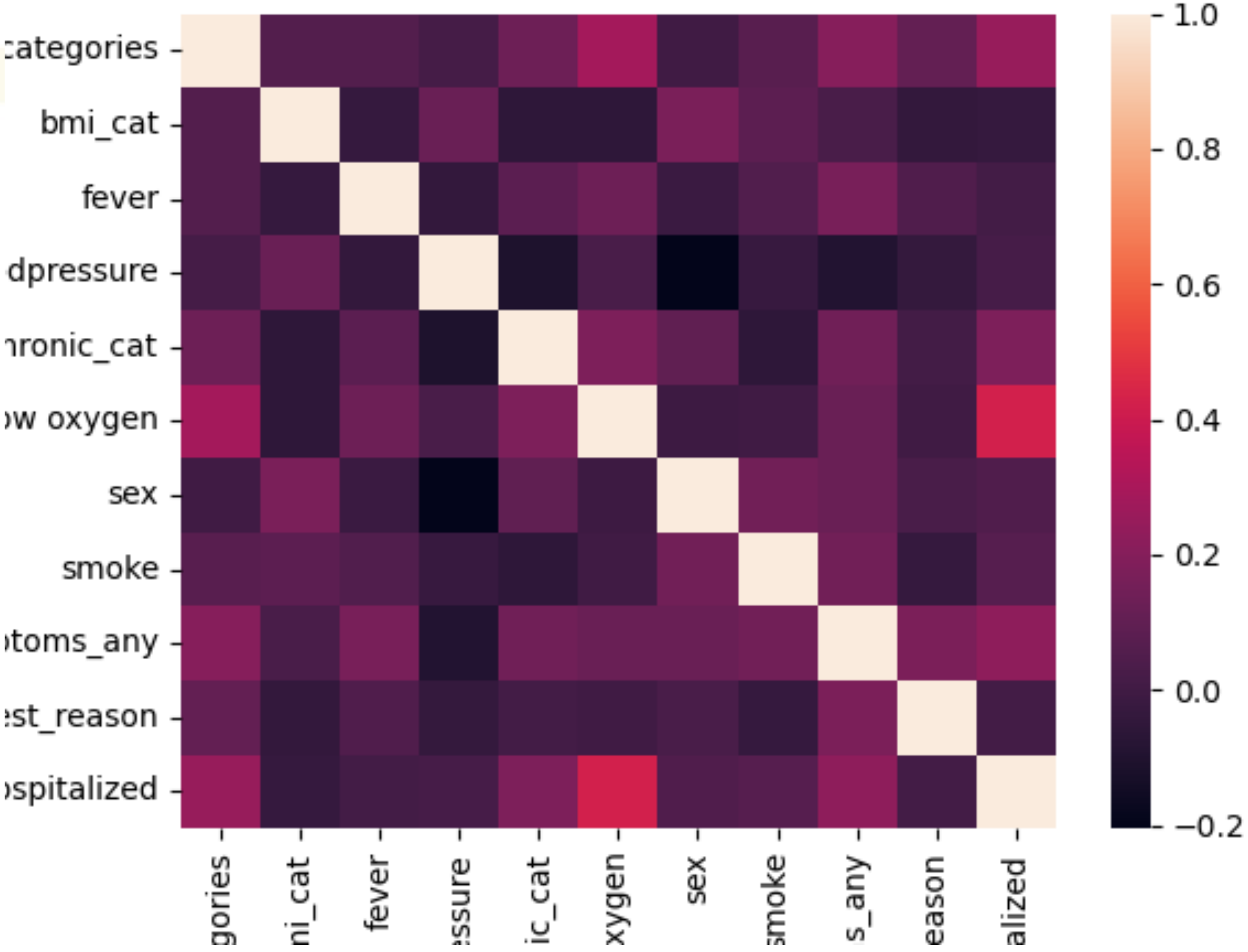
Distribution of the data



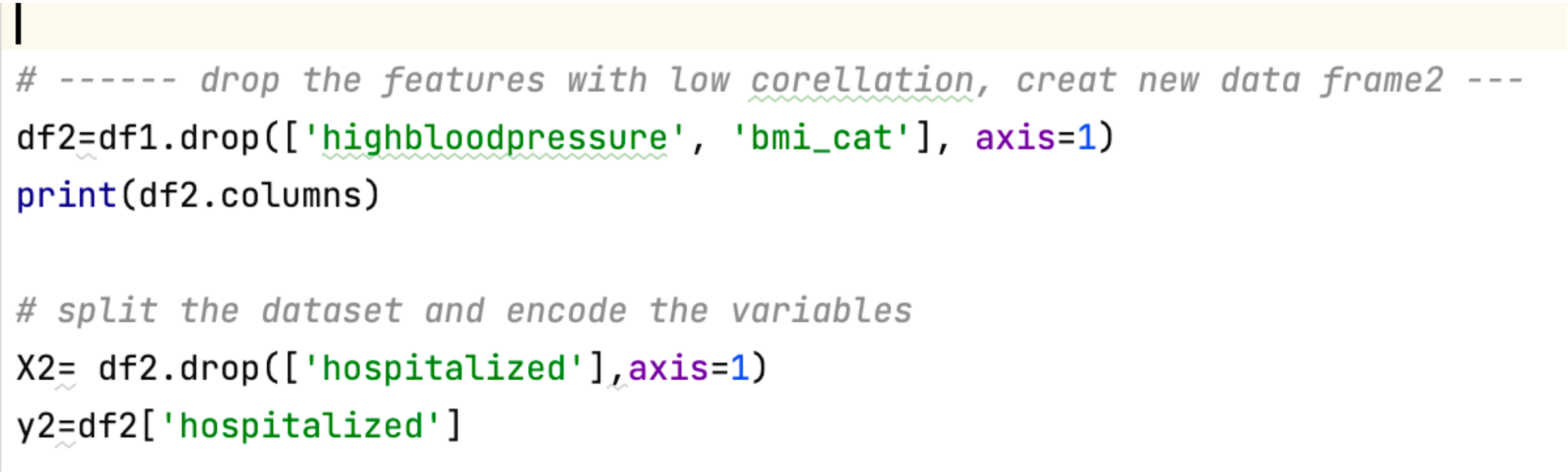
```
# check the distribution of the target
sns.displot(df1['hospitalized'])
plt.savefig('plot.png', dpi=300, bbox_inches='ti
plt.show()

# check correlation between the variables
sns.heatmap(df1.corr())
plt.show()
```





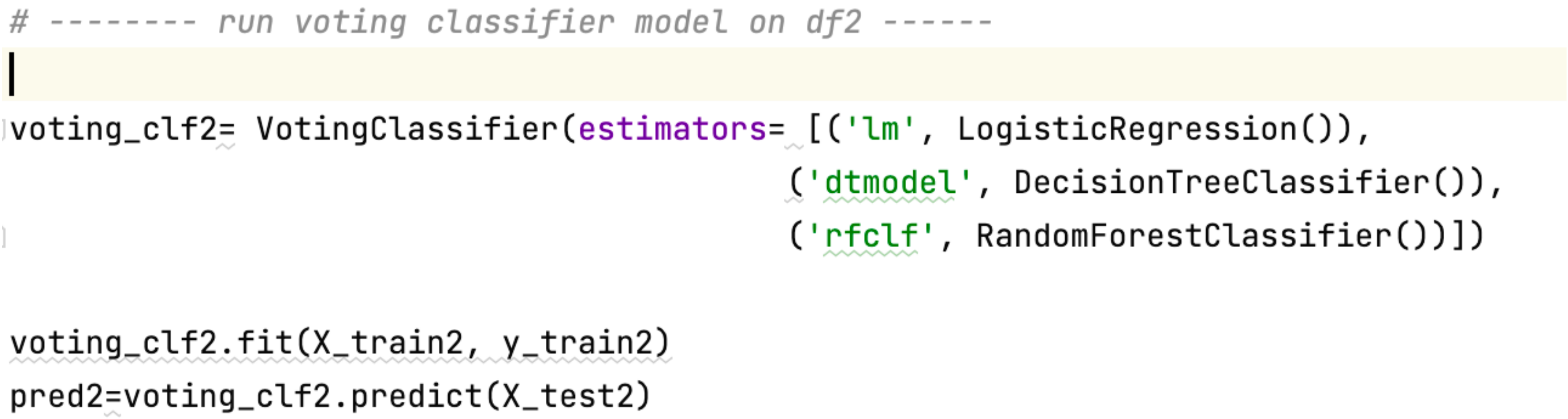
Drop features with low correlation



```
# ------ drop the features with low corellation, creat new data frame2 ---
df2=df1.drop(['highbloodpressure', 'bmi_cat'], axis=1)
print(df2.columns)

# split the dataset and encode the variables
X2= df2.drop(['hospitalized'],axis=1)
y2=df2['hospitalized']
```

Accuracy2 = 78.8

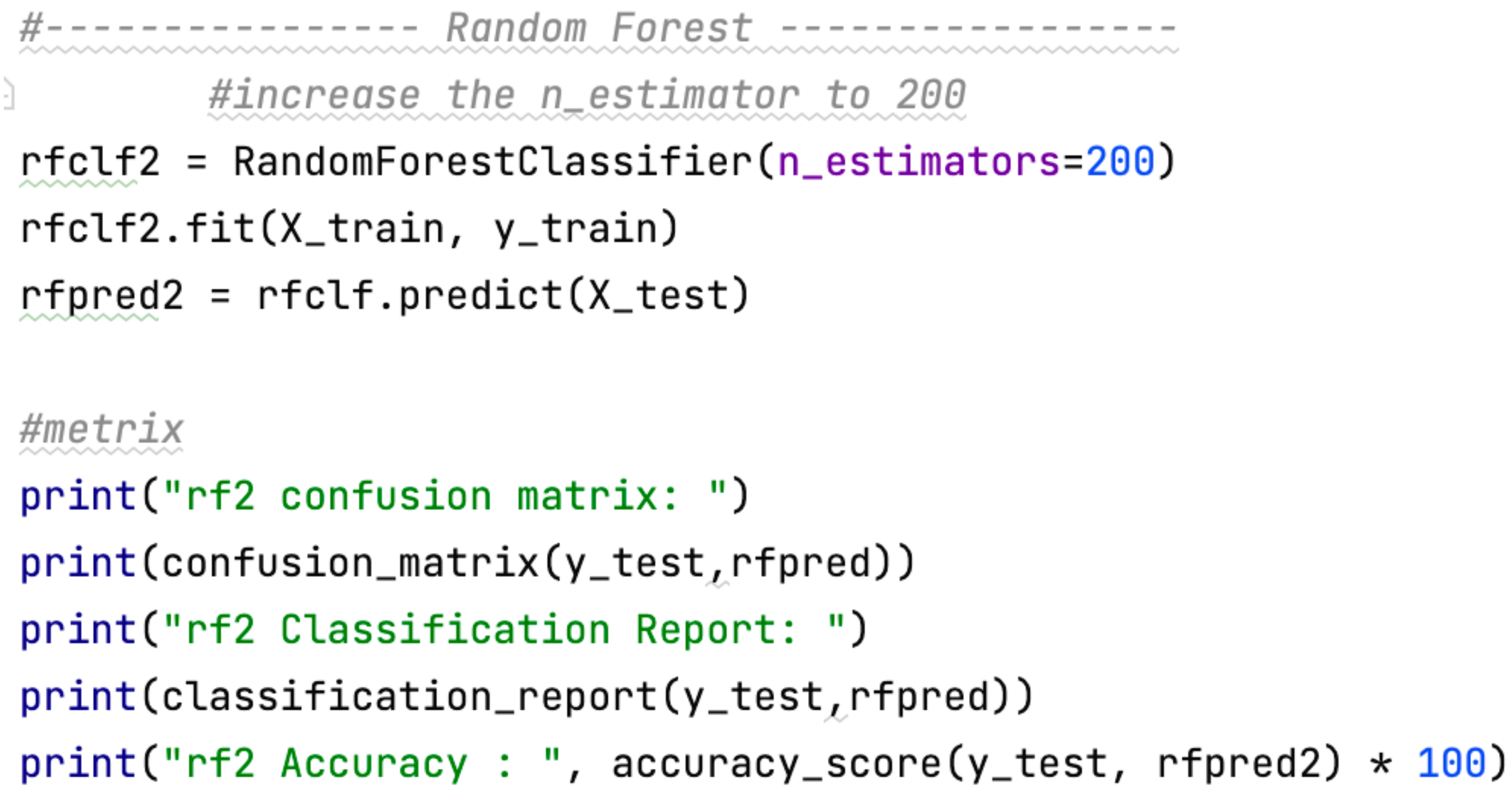


```
# -------- run voting classifier model on df2 ------

voting_clf2= VotingClassifier(estimators= [('lm', LogisticRegression()),
                                         ('dtmodel', DecisionTreeClassifier()),
                                         ('rfclf', RandomForestClassifier())])

voting_clf2.fit(X_train2, y_train2)
pred2=voting_clf2.predict(X_test2)
```

Optimize RF



```
#---------------- Random Forest -----------------
        #increase the n_estimator to 200
rfclf2 = RandomForestClassifier(n_estimators=200)
rfclf2.fit(X_train, y_train)
rfpred2 = rfclf.predict(X_test)

#metrix
print("rf2 confusion matrix: ")
print(confusion_matrix(y_test,rfpred))
print("rf2 Classification Report: ")
print(classification_report(y_test,rfpred))
print("rf2 Accuracy : ", accuracy_score(y_test, rfpred2) * 100)
```

Accuracy 78.8

Summary

- these three models have better accuracy in predicting the outcome

Logit reg		*True value*	
		0	1
Pred value	*0*	109	2
	1	27	18
Accuracy = 0.81			

SVM		*True value*	
		0	1
Pred value	*0*	108	3
	1	26	19
Accuracy = 0.81			

NB		*True value*	
		0	1
Pred value	*0*	99	12
	1	22	23
Accuracy = 0.81			

Thank you!

Questions