hw4 - lydiayu

March 29, 2022

[114]:
```python
import sklearn as sk
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import statsmodels.api as sm
```

0.0.1 Problem 1

[4]:
```python
heart = pd.read_csv("heart.csv")
```

[5]:
```python
heart
```

[5]:
```
      age  anaemia  creatinine_phosphokinase  diabetes  ejection_fraction  \
0    75.0        0                       582         0                 20
1    55.0        0                      7861         0                 38
2    65.0        0                       146         0                 20
3    50.0        1                       111         0                 20
4    65.0        1                       160         1                 20
..    …      …                       …       …                 …
294  62.0        0                        61         1                 38
295  55.0        0                      1820         0                 38
296  45.0        0                      2060         1                 60
297  45.0        0                      2413         0                 38
298  50.0        0                       196         0                 45

     high_blood_pressure  platelets  serum_creatinine  serum_sodium  sex  \
0                      1  265000.00               1.9           130    1
1                      0  263358.03               1.1           136    1
2                      0  162000.00               1.3           129    1
3                      0  210000.00               1.9           137    1
4                      0  327000.00               2.7           116    0
..                   …        …               …           …   …
294                    1  155000.00               1.1           143    1
295                    0  270000.00               1.2           139    0
```

```
296                        0  742000.00                 0.8            138    0
297                        0  140000.00                 1.4            140    1
298                        0  395000.00                 1.6            136    1

     smoking  time  DEATH_EVENT
0          0     4            1
1          0     6            1
2          1     7            1
3          0     7            1
4          0     8            1
..       …     …          …
294        1   270            0
295        0   271            0
296        0   278            0
297        1   280            0
298        1   285            0

[299 rows x 13 columns]
```

a)

[85]:
```python
# (median is value at 50th percentile in describe() function)
print(f"age\n{heart['age'].describe()} \n")
print(f"anaemia\n{heart['anaemia'].value_counts()} \n")
print(f"Creatinine Phosphokinase\n{heart['creatinine_phosphokinase'].
 ↪describe()}\n")
print(f"diabetes\n{heart['diabetes'].value_counts()}\n")
print(f"ejection fraction\n{heart['ejection_fraction'].describe()}\n")
print(f"high blood pressure\n{heart['high_blood_pressure'].value_counts()}\n")
print(f"mean platelets\n{heart['platelets'].describe()}\n")
print(f"serum creatinine\n{heart['serum_creatinine'].describe()}\n",)
print(f"serum sodium\n{heart['serum_sodium'].describe()}\n")
print(f"sex\n{heart['sex'].value_counts()}\n")
print(f"smoking\n{heart['smoking'].value_counts()}")
```

```
age
count    299.000000
mean      60.833893
std       11.894809
min       40.000000
25%       51.000000
50%       60.000000
75%       70.000000
max       95.000000
Name: age, dtype: float64

anaemia
```

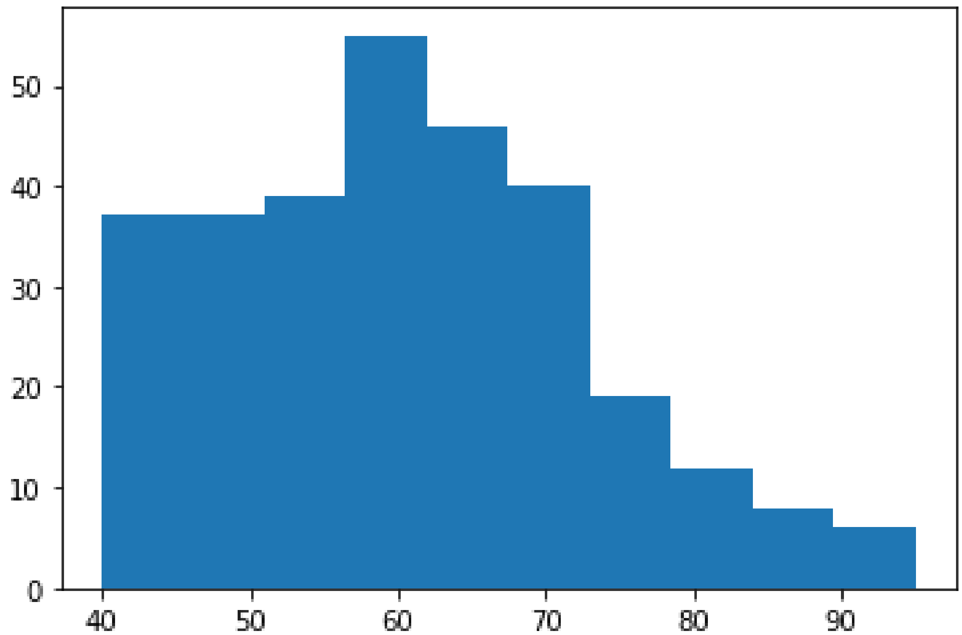
```
0    170
1    129
Name: anaemia, dtype: int64

Creatinine Phosphokinase
count     299.000000
mean      581.839465
std       970.287881
min        23.000000
25%       116.500000
50%       250.000000
75%       582.000000
max      7861.000000
Name: creatinine_phosphokinase, dtype: float64

diabetes
0    174
1    125
Name: diabetes, dtype: int64

ejection fraction
count    299.000000
mean      38.083612
std       11.834841
min       14.000000
25%       30.000000
50%       38.000000
75%       45.000000
max       80.000000
Name: ejection_fraction, dtype: float64

high blood pressure
0    194
1    105
Name: high_blood_pressure, dtype: int64

mean platelets
count       299.000000
mean     263358.029264
std       97804.236869
min       25100.000000
25%      212500.000000
50%      262000.000000
75%      303500.000000
max      850000.000000
Name: platelets, dtype: float64

serum creatinine
```

```
count    299.00000
mean       1.39388
std        1.03451
min        0.50000
25%        0.90000
50%        1.10000
75%        1.40000
max        9.40000
Name: serum_creatinine, dtype: float64

serum sodium
count    299.000000
mean     136.625418
std        4.412477
min      113.000000
25%      134.000000
50%      137.000000
75%      140.000000
max      148.000000
Name: serum_sodium, dtype: float64

sex
1    194
0    105
Name: sex, dtype: int64

smoking
0    203
1     96
Name: smoking, dtype: int64
```

b)

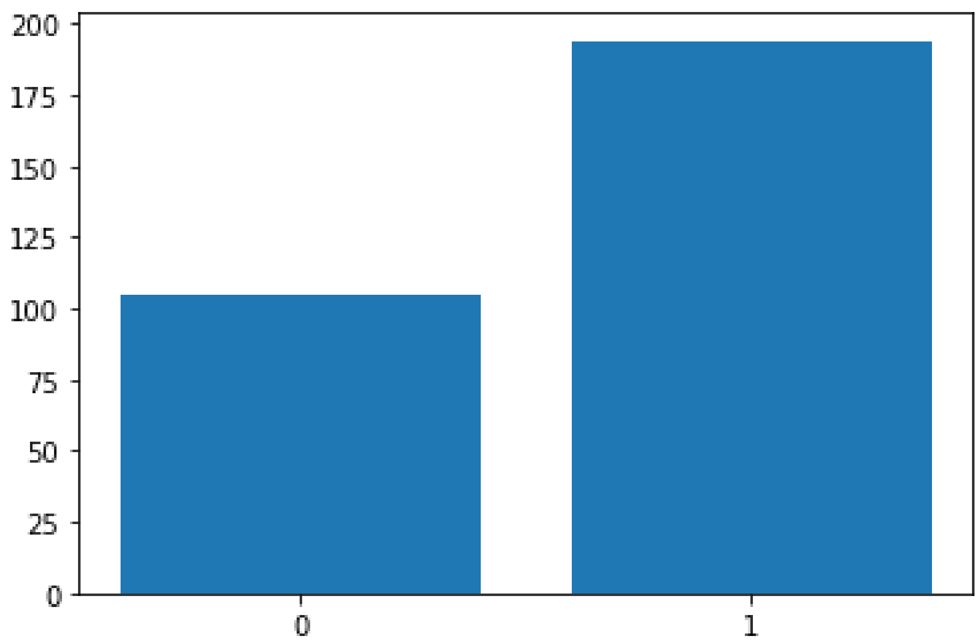
[27]:
```
# histogram of age
plt.hist(heart['age'])
```

[27]:
```
(array([37., 37., 39., 55., 46., 40., 19., 12.,  8.,  6.]),
 array([40. , 45.5, 51. , 56.5, 62. , 67.5, 73. , 78.5, 84. , 89.5, 95. ]),
 <BarContainer object of 10 artists>)
```



[40]:
```
# plot of sex
plt.bar(sorted(heart['sex'].unique()), sorted(heart['sex'].value_counts()),␣
↪tick_label=[0,1])
```

[40]: <BarContainer object of 2 artists>



The data does not seem very representative of the different values for sex – there are nearly twice as many observations where sex=1 than for sex=0. For age, there is not as much representation of individuals over age 70 – the counts for observations in this part of the histogram are much lower than the counts of observations under age 70.

c)

[51]:
```
# train test split
x_train, x_test, y_train, y_test = train_test_split(heart.loc[:, ~heart.columns.
↪isin(['time', 'DEATH_EVENT'])],
                                                    heart['DEATH_EVENT'],
                                                    test_size=0.3,
                                                    random_state=1234)
```

[61]:
```
# create model and fit with data
lr = LogisticRegression(solver='liblinear')
lr.fit(x_train, y_train)
```

[61]:
```
LogisticRegression(solver='liblinear')
```

[65]:
```
# see coefficients of model
lr.coef_[0]
pd.DataFrame(zip(x_train.columns, lr.coef_[0]), columns=['features', 'coef'])
```

[65]:

	features	coef
0	age	0.027412
1	anaemia	0.000452
2	creatinine_phosphokinase	0.000079
3	diabetes	0.000105
4	ejection_fraction	-0.032819
5	high_blood_pressure	0.000310
6	platelets	0.000001
7	serum_creatinine	0.004858
8	serum_sodium	-0.012412
9	sex	0.000013
10	smoking	-0.000004

The coefficient for `smoking` is 0. This implies that smoking has no effect on the probability of whether or not a patient dies of heart failure.

d)

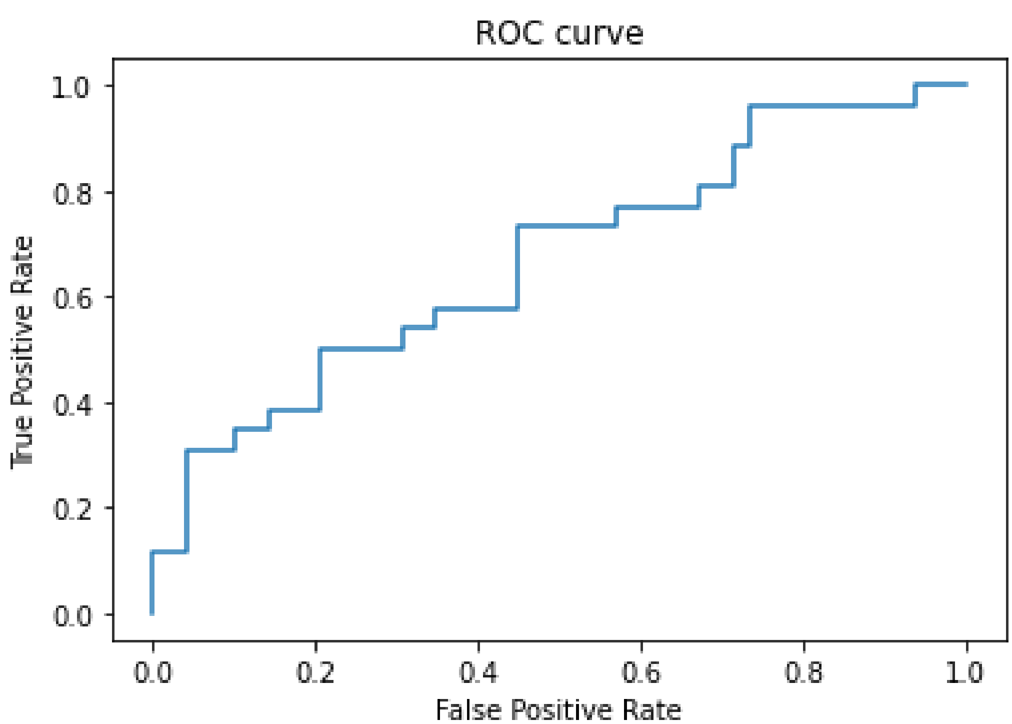
[96]:
```
lr.classes_
```

[96]:
```
array([0, 1])
```

[100]:
```
# predict probabilities of death for each observation in x_test
# predict_proba returns an nx2 array, each row represents probability of death␣
↪= 0 and death = 1
# we just want the probabilities of death = 1 (second element of each row in␣
↪array)
y_pred_proba = lr.predict_proba(x_test)[:,1]
```

[86]:
```
# get FP rate & TP rate given thresholds and threshold values
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_proba)
```

[101]:
```
# plot ROC curve
plt.plot(fpr,tpr)
plt.title("ROC curve")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



[73]:
```
# calculate auc
auc = metrics.roc_auc_score(y_test, y_pred_proba)
auc
```

[73]: 0.6695447409733123

Many recent models use random forest to predict heart failure mortality. These have a C-statistic

(AUC) of around 0.7, which is slightly better than this model. The model can be improved by increasing the number of features – many of the most recent papers written on heart failure mortality involve models with orders of magnitude more features than this one. Also, a different classifier other than logistic regression could be used, such as a decision tree or support vector machine, which can handle more complex decision boundaries.

0.0.2 Problem 2

[106]:
```
climate = pd.read_csv("climate_change.csv")
```

[107]:
```
climate
```

[107]:
```
     Year  Month    MEI     CO2      CH4      N2O   CFC-11   CFC-12  \
0    1983      5  2.556  345.96  1638.59  303.677  191.324  350.113
1    1983      6  2.167  345.52  1633.71  303.746  192.057  351.848
2    1983      7  1.741  344.15  1633.22  303.795  192.818  353.725
3    1983      8  1.130  342.25  1631.35  303.839  193.602  355.633
4    1983      9  0.428  340.17  1648.40  303.901  194.392  357.465
..    …      …     …       …        …        …        …        …
303  2008      8 -0.266  384.15  1779.88  321.405  244.200  535.072
304  2008      9 -0.643  383.09  1795.08  321.529  244.083  535.048
305  2008     10 -0.780  382.99  1814.18  321.796  244.080  534.927
306  2008     11 -0.621  384.13  1812.37  322.013  244.225  534.906
307  2008     12 -0.666  385.56  1812.88  322.182  244.204  535.005

           TSI  Aerosols   Temp
0    1366.1024    0.0863  0.109
1    1366.1208    0.0794  0.118
2    1366.2850    0.0731  0.137
3    1366.4202    0.0673  0.176
4    1366.2335    0.0619  0.149
..         …        …      …
303  1365.6570    0.0036  0.407
304  1365.6647    0.0043  0.378
305  1365.6759    0.0046  0.440
306  1365.7065    0.0048  0.394
307  1365.6926    0.0046  0.330

[308 rows x 11 columns]
```

a)

[126]:
```
# training data is everything up to and including 2006
X = climate.loc[:, ~ climate.columns.isin(['Temp'])]
Y = climate[['Year', 'Temp']]
```

```python
X_train = X[X['Year'] <= 2006].drop(columns=['Year', 'Month'])
# include intercept
X_train = sm.add_constant(X_train)
X_test = X[X['Year'] > 2006].drop(columns=['Year', 'Month'])
Y_train = Y[Y['Year'] <= 2006].drop(columns=['Year'])
Y_test = Y[Y['Year'] > 2006].drop(columns=['Year'])
```

[127]:
```python
# build linear regression model
model = sm.OLS(Y_train, X_train).fit()
```

[128]:
```python
model.summary()
```

[128]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Temp   R-squared:                       0.751
Model:                            OLS   Adj. R-squared:                  0.744
Method:                 Least Squares   F-statistic:                     103.6
Date:                Thu, 17 Mar 2022   Prob (F-statistic):           1.94e-78
Time:                        22:22:37   Log-Likelihood:                 280.10
No. Observations:                 284   AIC:                            -542.2
Df Residuals:                     275   BIC:                            -509.4
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -124.5943     19.887     -6.265      0.000    -163.744     -85.445
MEI            0.0642      0.006      9.923      0.000       0.051       0.077
CO2            0.0065      0.002      2.826      0.005       0.002       0.011
CH4            0.0001      0.001      0.240      0.810      -0.001       0.001
N2O           -0.0165      0.009     -1.930      0.055      -0.033       0.000
CFC-11        -0.0066      0.002     -4.078      0.000      -0.010      -0.003
CFC-12         0.0038      0.001      3.757      0.000       0.002       0.006
TSI            0.0931      0.015      6.313      0.000       0.064       0.122
Aerosols      -1.5376      0.213     -7.210      0.000      -1.957      -1.118
==============================================================================
Omnibus:                        8.740   Durbin-Watson:                   0.956
Prob(Omnibus):                  0.013   Jarque-Bera (JB):               10.327
Skew:                           0.289   Prob(JB):                      0.00572
Kurtosis:                       3.733   Cond. No.                     8.53e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
[2] The condition number is large, 8.53e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

The R^2 for this model is 0.751.

b) The variables that are significant are `MEI`, `CO2`, `CFC-11`, `CFC-12`, and `Aerosols`. The coefficients for `N2O` and `CFC-11` are negative, implying that the more of these molecules in the atmosphere, the lower `Temp` is, which is counter-intuitive. This is likely due to collinearity with other variables in the model.

c)

[118]:
```python
# correlation matrix
climate.loc[:, ~climate.columns.isin(['Temp', 'Year', 'Month'])].corr()
```

[118]:

	MEI	CO2	CH4	N2O	CFC-11	CFC-12
MEI	1.000000	-0.152911	-0.105555	-0.162375	0.088171	-0.039836
CO2	-0.152911	1.000000	0.872253	0.981135	0.401284	0.823210
CH4	-0.105555	0.872253	1.000000	0.894409	0.713504	0.958237
N2O	-0.162375	0.981135	0.894409	1.000000	0.412155	0.839295
CFC-11	0.088171	0.401284	0.713504	0.412155	1.000000	0.831381
CFC-12	-0.039836	0.823210	0.958237	0.839295	0.831381	1.000000
TSI	-0.076826	0.017867	0.146335	0.039892	0.284629	0.189270
Aerosols	0.352351	-0.369265	-0.290381	-0.353499	-0.032302	-0.243785

	TSI	Aerosols
MEI	-0.076826	0.352351
CO2	0.017867	-0.369265
CH4	0.146335	-0.290381
N2O	0.039892	-0.353499
CFC-11	0.284629	-0.032302
CFC-12	0.189270	-0.243785
TSI	1.000000	0.083238
Aerosols	0.083238	1.000000

`N2O` is highly correlated with `CO2`, `CH4`, and `CFC-12`.

d) To avoid the problem with multicollinearity, re-build the model excluding the variables that are highly correlated with other variables in the model (in this case, `N2O` and `CFC-11`, although it is also arguable that `CO2`, `CH4`, and `CFC-12` are also highly correlated with reach other).

[123]:
```python
# training test split, excluding N2O and CFC-11
X = climate.loc[:, ~ climate.columns.isin(['Temp', 'N2O', 'CFC-11'])]
Y = climate[['Year', 'Temp']]
```

```
X_train = X[X['Year'] <= 2006].drop(columns=['Year', 'Month'])
X_train = sm.add_constant(X_train)
X_test = X[X['Year'] > 2006].drop(columns=['Year', 'Month'])
Y_train = Y[Y['Year'] <= 2006].drop(columns=['Year'])
Y_test = Y[Y['Year'] > 2006].drop(columns=['Year'])
```

[124]:
```
# build linear regression model
model2 = sm.OLS(Y_train, X_train).fit()
```

[125]:
```
model2.summary()
```

[125]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Temp   R-squared:                       0.734
Model:                            OLS   Adj. R-squared:                  0.728
Method:                 Least Squares   F-statistic:                     127.3
Date:                Thu, 17 Mar 2022   Prob (F-statistic):           1.25e-76
Time:                        22:21:54   Log-Likelihood:                 270.72
No. Observations:                 284   AIC:                            -527.4
Df Residuals:                     277   BIC:                            -501.9
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -118.0132     20.411     -5.782      0.000    -158.194     -77.833
MEI            0.0623      0.007      9.371      0.000       0.049       0.075
CO2            0.0104      0.001      9.712      0.000       0.008       0.012
CH4            0.0002      0.001      0.468      0.640      -0.001       0.001
CFC-12        -0.0001      0.000     -0.315      0.753      -0.001       0.001
TSI            0.0836      0.015      5.592      0.000       0.054       0.113
Aerosols      -1.5844      0.219     -7.236      0.000      -2.015      -1.153
==============================================================================
Omnibus:                       14.634   Durbin-Watson:                   0.897
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               17.878
Skew:                           0.435   Prob(JB):                     0.000131
Kurtosis:                       3.869   Cond. No.                     8.38e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 8.38e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

The new R^2 value is 0.734.

```
[ ]:
```