Uczenie maszynowe

P.Chajduła, P.Szewerniak, K.Oratowski

December 2020

1 Wstep

Naszym zadaniem było sprawdzenie stopnia poprawności przewidywania różnych modeli uczenia maszynowego oraz wdrożenie najlepszego z tychże w celu ustalenia, które czynniki mają znaczący wpływ na wystąpienie u pacjenta choroby serca. Finalnie otrzymaliśmy wartości procentowe zgodności różnych podejść do naszego zestawu danych oraz wyniki zastosowania lasu losowego i regresji liniowej.

2 Metoda badawcza

Zaczynamy od zaimportowania potrzebnych bibliotek oraz danych i sprawdzenia ich integralności tzn. czy w tabeli są wszystkie dane, czy są to dane liczbowe, czy nie ma niepoprawnych wartości.

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
from pandas import ExcelWriter
```

```
from pandas import ExcelFile
```

```
df = pd.read_csv('heart.csv')
```

```
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
df.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
df.isnull().values.any()
```

```
False
```

```
df1=df.sort_values(by=['target'])
```

```
df1
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0
200	44	1	0	110	197	0	0	177	0	0.0	2	1	2	0
201	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
202	58	1	0	150	270	0	0	111	1	0.8	2	0	3	0
203	68	1	2	180	274	1	0	150	1	1.6	1	0	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
98	43	1	2	130	315	0	1	162	0	1.9	2	1	2	1
97	52	1	0	108	233	1	1	147	0	0.1	2	3	3	1
96	62	0	0	140	394	0	0	157	0	1.2	1	0	2	1
94	45	0	1	112	160	0	1	138	0	0.0	1	0	2	1
151	71	0	0	112	149	0	1	125	0	1.6	1	0	2	1

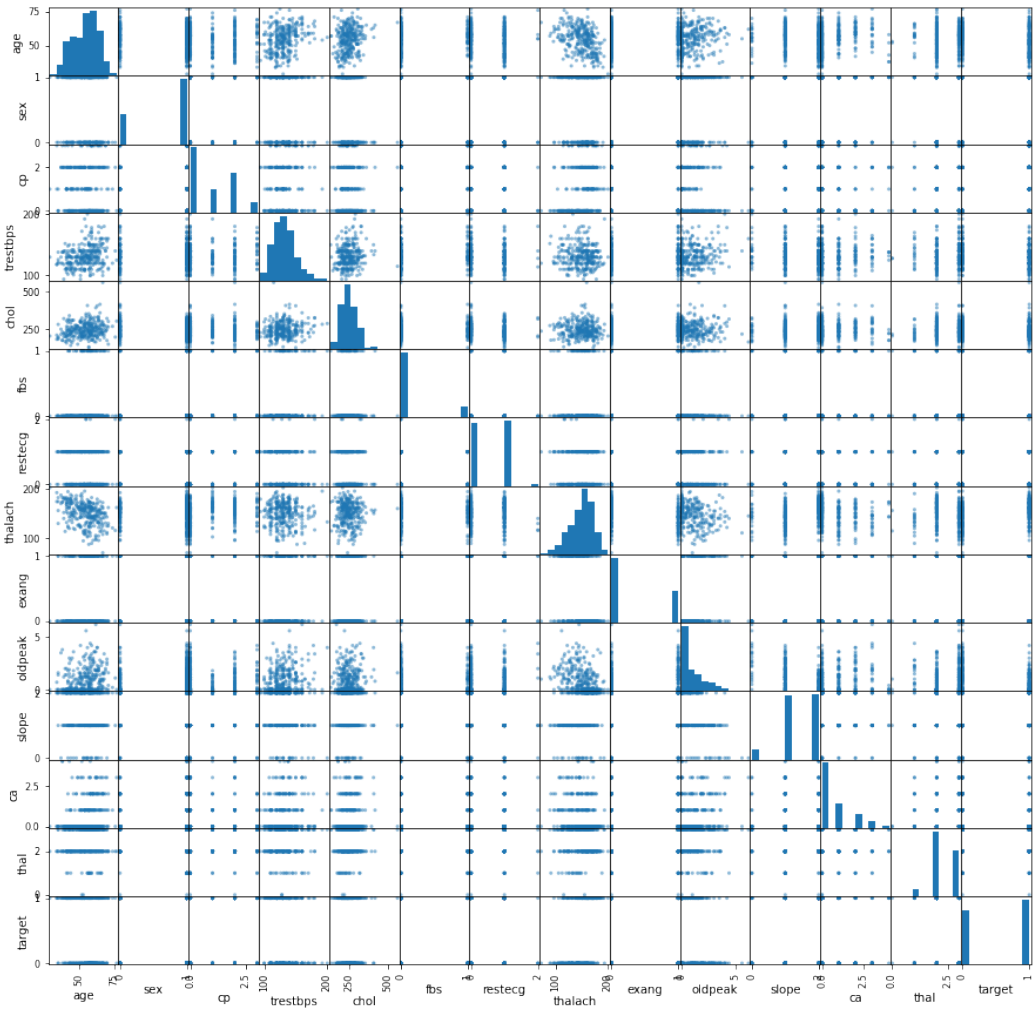
303 rows × 14 columns

Wyjaśnienie nazw: age - wiek, sex - płeć, cp - ból w klatce piersiowej - skala od 0 do 3, trestbps - spoczynkowe ciśnienie krwi, chol - stężenie cholersterolu, fbs - czy poziom cukru przekracza 120mg/dl, restecg - spoczynkowe ekg - skala od 0 do 2, gdzie 0 to brak odchyleń od normy, thalach - maksymalne tętno, exang - czy pacjent przebył niedawno anginę, oldpeak - czy wysiłek wpływa na kształt odcinka ST w ekg, slope - typ odcinka ST, ca - wynik fluoroskopii , thal - badanie thalem, target - czy u pacjenta wykryto chorobę serca

Tworzymy macierz kowariancji żeby stwierdzić, które dane są ze sobą skorelowane

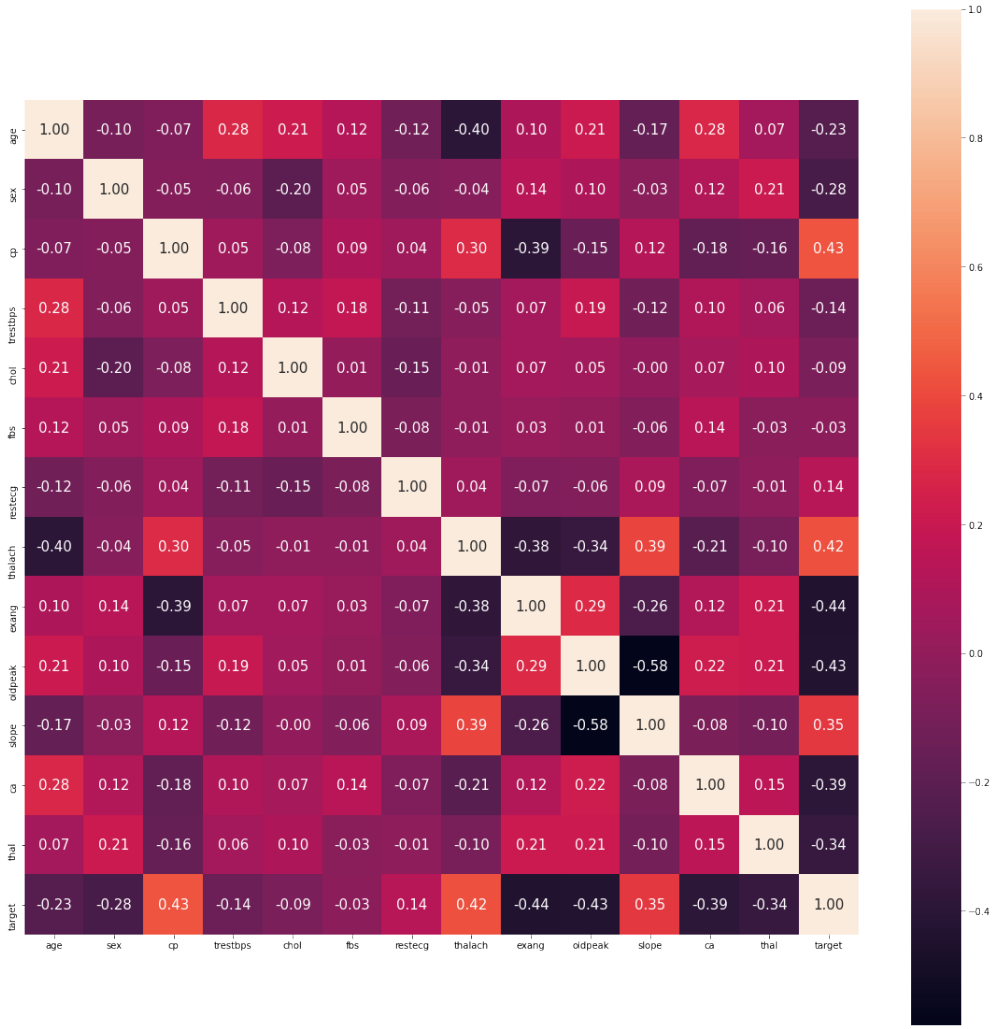
```
def cov_matrix(m):
    print(m.columns)
    cov_data = np.corrcoef(m.T);
    plt.figure(figsize=(20,20))
    sns.heatmap(cov_data, cbar=True, annot=True, square=True, annot_kws={'size':15}, fmt='.2f', xticklabels= m.columns, yticklabe
    return(cov_data)
```

```
mScatterPlot=pd.plotting.scatter_matrix(df1, alpha =0.5, figsize=(15,15), grid=True)
```




```
mCorMatrix = cov_matrix(df1)
```

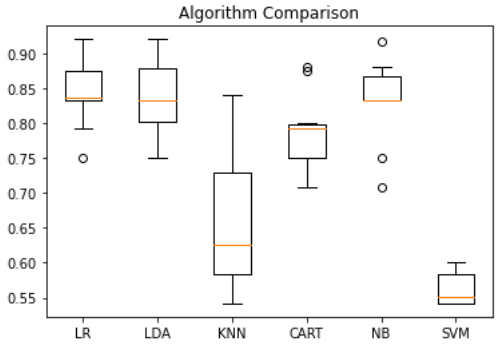
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```



Z otrzymanej powyżej mapy widać znaczącą dodatnią korelacje między cp a target,thalach a target,slope a thalach. Znacząca korelacja ujemna występuję między slope a oldpeak,exang a target.

W tym miejscu zaczynamy się zastanawiać który model byłby najlepszy dla naszych danych. W tym celu wykorzystujemy bilbiotekę sklearn która skraca nam pracę dzięki wbudowanym metodom.

```
In [1]: from pandas import read_csv
        from matplotlib import pyplot
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import cross_val_score
        from sklearn.model_selection import StratifiedKFold
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
        from sklearn.naive_bayes import GaussianNB
        from sklearn.svm import SVC

In [2]: dataset = read_csv('heart.csv')

In [3]: array = dataset.values

In [5]: X = array[:,0:13]

In [7]: y = array[:,13]

In [9]: X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20, random_state=1, shuffle=True)

In [10]: models = []

In [11]: models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
         models.append(('LDA', LinearDiscriminantAnalysis()))
         models.append(('KNN', KNeighborsClassifier()))
         models.append(('CART', DecisionTreeClassifier()))
         models.append(('NB', GaussianNB()))
         models.append(('SVM', SVC(gamma='auto')))

In [12]: results = []
         names = []

In [13]: for name, model in models:
             kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
             cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
             results.append(cv_results)
             names.append(name)
             print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

         LR: 0.842667 (0.045358)
         LDA: 0.842500 (0.052864)
         KNN: 0.651667 (0.095948)
         CART: 0.788833 (0.051855)
         NB: 0.830333 (0.057777)
         SVM: 0.561833 (0.022091)

In [14]: pyplot.boxplot(results, labels=names)
         pyplot.title('Algorithm Comparison')
```



Na powyższym obrazku mamy przedstawione wyniki dokładności wybranych modelów (LR - regresja liniowa, LDA - liniowa analiza dyskryminacyjna, KNN - najbliżsi sąsiedzi, CART - drzewo decyzyjne, NB - naiwny klasyfikator bayesowski, SVM - maszyna wektorów wspierajacych) w sposób graniczny. Widzimy, że najlepszym modelem dla naszych danych będzie model LR tzn. model regresji liniowej.

```
model = LogisticRegression(solver='liblinear', multi_class='ovr')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

```
print(accuracy_score(Y_validation, predictions))
```

```
0.7540983606557377
```

Wykorzystując gotowy model regresji liniowej z biblioteki sklearn, sprawdzamy w jakim procencie prognozy są poprawne. Jak widzimy zastosowany przez nas model ma dokładność rzędu 75%.

Postanowiliśmy również sprawdzić model lasu losowego, ponieważ wcześniej nie braliśmy tego modelu pod uwagę i ku naszemu zaskoczeniu model lasu losowego był minimalnie lepszy co przedstawiamy poniżej(dokładnośc tego modelu wyniosła 76%)

```
labels=np.array(df1['target'])
```

```
features=df1.drop('target',axis=1)
```

```
features
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2
200	44	1	0	110	197	0	0	177	0	0.0	2	1	2
201	60	1	0	125	258	0	0	141	1	2.8	1	1	3
202	58	1	0	150	270	0	0	111	1	0.8	2	0	3
203	68	1	2	180	274	1	0	150	1	1.6	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
98	43	1	2	130	315	0	1	162	0	1.9	2	1	2
97	52	1	0	108	233	1	1	147	0	0.1	2	3	3
96	62	0	0	140	394	0	0	157	0	1.2	1	0	2
94	45	0	1	112	160	0	1	138	0	0.0	1	0	2
151	71	0	0	112	149	0	1	125	0	1.6	1	0	2

303 rows × 13 columns

```
feature_list=list(features.columns)
```

```
features = np.array(features)
```

```
from sklearn.model_selection import train_test_split
```

```
train_features, test_features,train_labels, test_labels = train_test_split(features,labels,test_size=0.75,random_state=1)
```

```
print('Training Features Shape:',train_features.shape)
print('Training Labels Shape:',train_labels.shape)
print('Testing Features Shape:',test_features.shape)
print('Testing Labels Shape:',test_labels.shape)
```

```
Training Features Shape: (75, 13)
Training Labels Shape: (75,)
Testing Features Shape: (228, 13)
Testing Labels Shape: (228,)
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
rf=RandomForestRegressor(n_estimators=1000,random_state=1)
```

```
rf.fit(train_features,train_labels)
```

```
RandomForestRegressor(n_estimators=1000, random_state=1)
```

```
predictions=rf.predict(test_features)
```

```
len(test_labels)
```

```
228
```

```
len(predictions)
```

```
228
```

```
rounded_predictions=predictions.round(0)
```

```
int_predictions=rounded_predictions.astype(int)
```

```
equal=test_labels==int_predictions
```

```
occurences=np.count_nonzero(equal==True)
```

```
print(occurences)
```

```
175
```

```
occurences/len(predictions)
```

```
0.7675438596491229
```

3 Podsumowanie

Po obróbce i przygotowaniu danych związanych z czynnikami wpływającymi na chorobę serca, sprawdziliśmy liniowe zależności pomiędzy zmiennymi. Następnie wybraliśmy z sześciu różnych modeli najbardziej odpowiedni i okazał się nim być model regresji liniowej. Po zastosowaniu modelu otrzymaliśmy sprawność prognozowania choroby serca na poziomie 75%. Stwierdziliśmy jednak że warto róznież zastosować model lasu losowego którego dokładność przewidywania była minimalnie lepsza od modelu regresji liniowej.