Projekt 1

Michał Dąbrowski

14 kwietnia 2021

Spis treści

1 Dane

Przedmiotem badań będzie sprawdzenie jak pięć metod klasyfikacji danych poradzi sobie z przewidzeniem jakości wina na podstawie 12 zmiennych, którymi są:

- Fixed acidity - kwasowosc stala
- volatile acidity - kwasowosc lotna
- citric acid - kwas cytrynowy
- residual sugar - cukier resztkowy
- chlorides - chlorki
- free sulfur dioxide - wolny dwutlenek siarki
- total sulfur dioxide - całkowity dwutlenek siarki
- density - gęstość
- pH
- sulphates - siarczany
- alcohol - zawartość alkoholu
- quality - jakość

2 Czyszczenie bazy danych

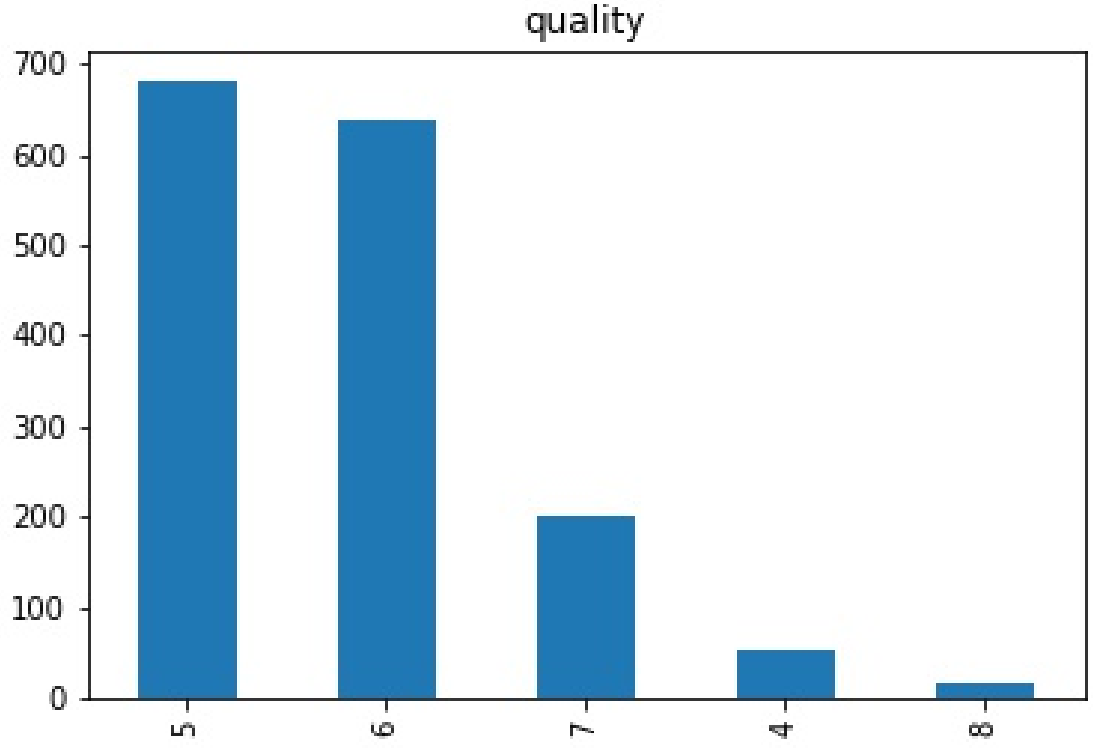
Przed przystąpieniem do klasyfikacji sprawdziłem czy w bazie danych znajdują się zmienne inne niż liczbowe za pomocą kodu : Nie otrzymałem żadnego komunikatu

```
def sprawdzajaca(df):
    for i in df.columns :
        if(df[i].dtypes == 'int64' or df[i].dtypes == 'float64'):
            pass
        else:
            print(df[i][0:10])
sprawdzajaca(df)
```

w konsoli co oznacza, że wszystkie zmienne są zmiennymi liczbowymi. Następnie sprawdziłem czy, któraś z kolumn nie zawiera wartości NA i okazało się, że baza nie zawiera tych wartości.

3 Wykresy

Zrobiłem wykresy z wartości minimalnej, maksymalnej i średniej dla wszystkich kolumn, jak również wykresy najczęściej pojawiających się wartości jednak nie wskazują one nic co mogło by mieć znaczenie przy klasyfikacji. Jedynie wykres jakości wina ma tu znaczenie, ponieważ widać na nim, że najczęściej wina miały jakość 5 lub 6 i stanowiło to ponad 80 % wszystkich wyników.

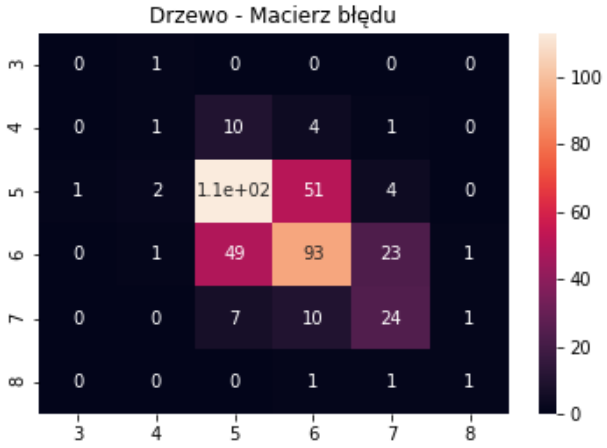


Pozostałe wykresy umieściłem na końcu pracy jako ciekawostkę.

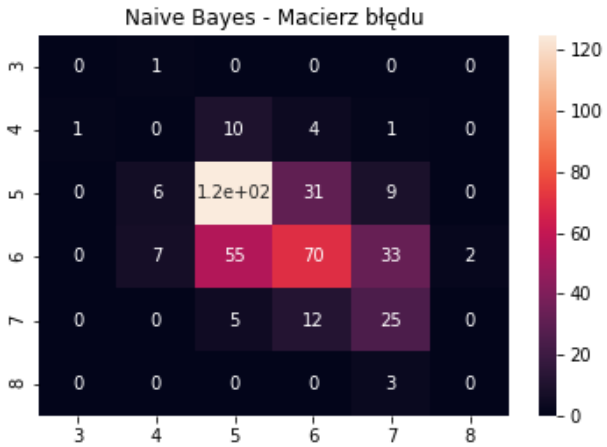
4 Klasyfikatory

Użyłem 5 klasyfikatorów :

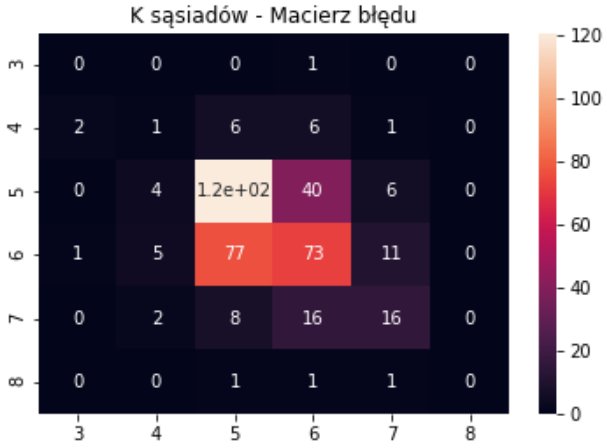
- Drzewa decyzyjnego
  Uzyskałem tu skuteczność na poziomie około 50-60 %, a macierz błędu wygląda następująco:



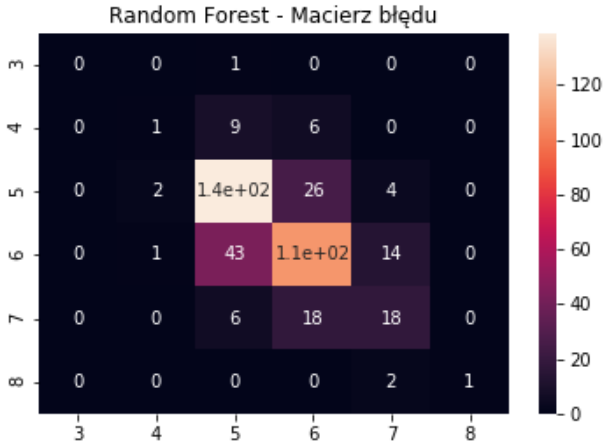
- Naive Bayes Uzyskałem tu skuteczność na poziomie około 55 %, a macierz błędu wygląda następująco:



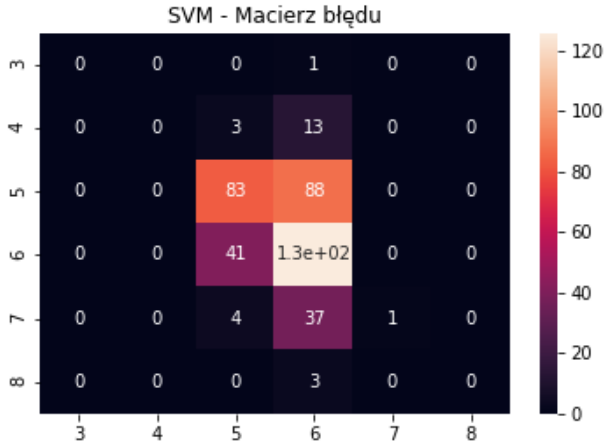
- K sąsiadów Uzyskałem tu skuteczność na poziomie około 52 %, a macierz błędu wygląda następująco:



- Random Forest Uzyskałem tu skuteczność na poziomie około 65-70 %, a macierz błędu wygląda następująco:



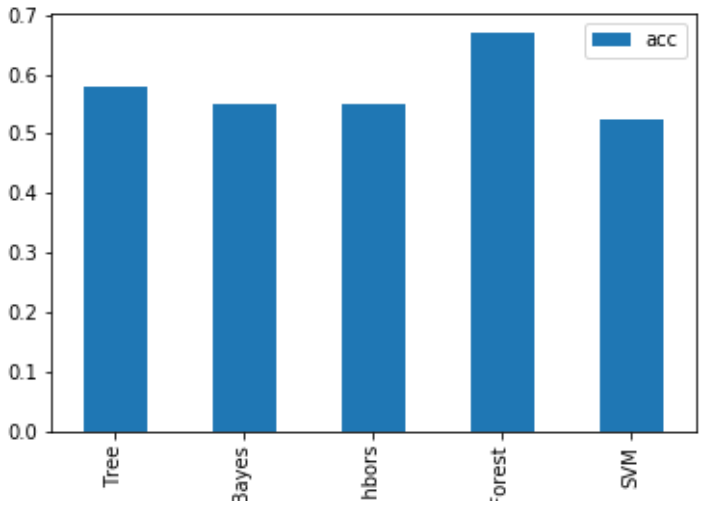
- SVC Uzyskałem tu skuteczność na poziomie około 52 %, a macierz błędu wygląda następująco:



Od strony kodu wszystkie klasyfikatory wyglądają tak samo wyłącznie ze zmianą używanej funkcji klasyfikatora, dla tego pokażę kod tylko jednego klasyfikatora.

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Wyniki z random forest: ",round(clf.score(X_test,y_test)*100,2),'%')
acc.append(clf.score(X_test,y_test))
print(confusion_matrix(y_test, y_pred))

sn.heatmap(confusion_matrix(y_test, y_pred),xticklabels = xl , yticklabels = xl , annot=True)
plt.title('Random Forest - Macierz błędu')
plt.savefig('Random Forest - Macierz błędu')
plt.show()
```

Po zebraniu wszystkich dokładności klasyfikatorów otrzymałem następujący wykres:



5 Wnioski

Po analizie wszystkich macierzy błędu widać, że większość klasyfikatorów błędnie przyporządkowuje wino do jakości 5 lub 6, co jest efektem tego, że w bazie danych

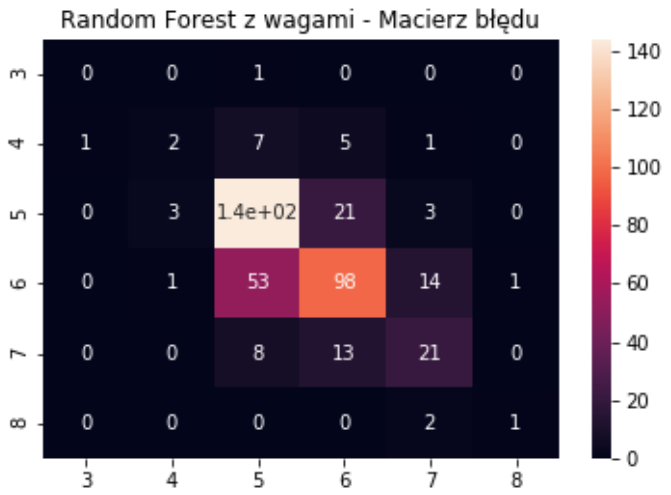
znajdowało się wiele rzędów z wartościami 5 lub 6 w zmiennej jakości.
Widać również ,że najlepiej ze wszystkich klasyfikatorów radził sobie ”Random Forest”, który działa w oparciu o drzewo decyzyjne natomiast tworzy on kilka drzew decyzyjnych w czasie uczenia się klasyfikatora i wybiera najlepiej pasujące.

6 Skalowanie wartości

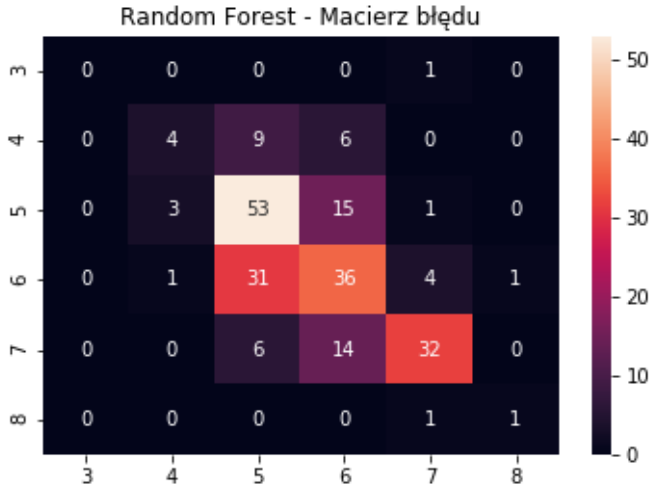
Biorąc pod uwagę wnioski najlepszym klasyfikatorem okazuje się klasyfikator ”Random Forest”, natomiast największym problemem bazy danych jest jej brak balansu. Można to obejść za pomocą nadawania większego znaczenia jednym danym (np. jakość wina określana jako 3,4,7,8 będzie dużo ważniejsza przy uczeniu się programu niż 5 i 6).

```
val = []
for i in df['quality'].value_counts().to_list():
    val.append((i/df['quality'].count())**(-1))

clf = RandomForestClassifier(n_estimators=10,class_weight = {3: val[-1], 4: val[-3], 5: val[-6], 6: val[-5], 7:val[-4], 8:val[-2]})
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Wyniki z random forest: ",round(clf.score(X_test,y_test)*100,2),'%')
acc.append(clf.score(X_test,y_test))
print(confusion_matrix(y_test, y_pred))

sn.heatmap(confusion_matrix(y_test, y_pred),xticklabels = xl , yticklabels = xl , annot=True)
plt.title('Random Forest - Macierz błędu')
plt.savefig('Random Forest - Macierz błędu')
plt.show()
```

Jednak pomimo zmiany znaczenia jakości wina 5 i 6 wciąż otrzymuję podobne wyniki.



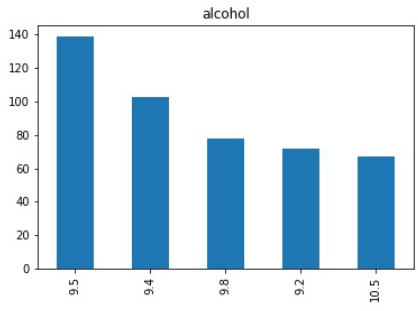
Spróbowałem również pomniejszyć liczbę rzędów w których jakość była równa 5 i 6 jednak to sprawiło, że dokładność klasyfikatora spadła i nie wynosiła więcej niż 55 %.



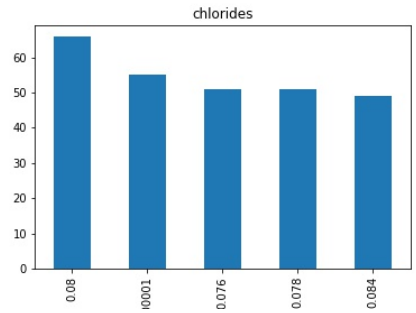
7 Wniosek

Powyższe klasyfikatory nie były w stanie poradzić sobie z tą bazą danych prawdopodobnie ze względu na dużą zależność pomiędzy danymi.

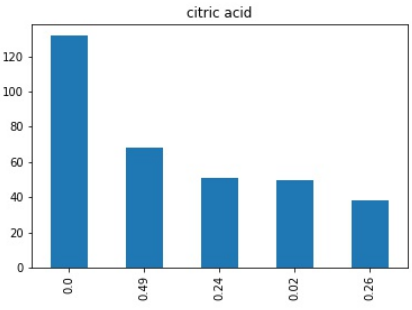
8 Wszystkie wykresy



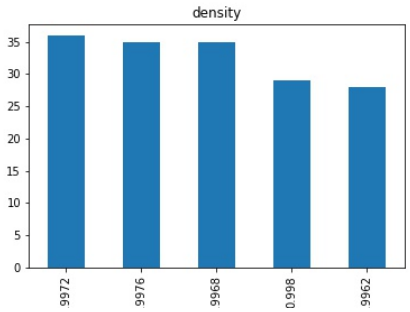
(a)



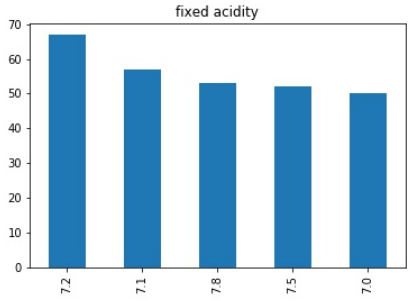
(b)


citric acid
120
100
80
60
40
20
0
0.0
0.49
0.24
0.02
0.26

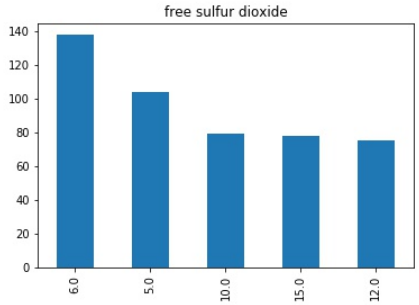
(a)


density
35
30
25
20
15
10
5
0
9972
9976
9968
0.998
9962

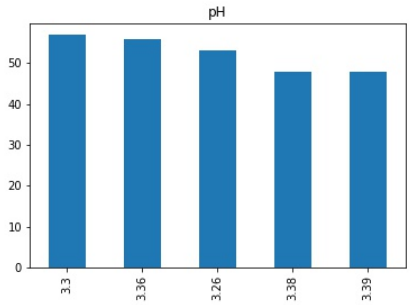
(b)


fixed acidity
70
60
50
40
30
20
10
0
7.2
7.1
7.8
7.5
7.0

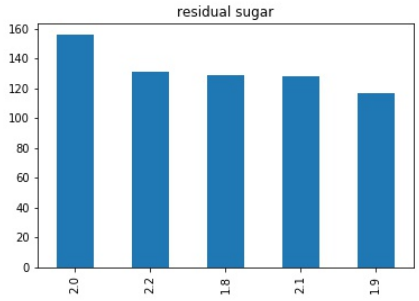
(a)


free sulfur dioxide
140
120
100
80
60
40
20
0
6.0
5.0
10.0
15.0
12.0

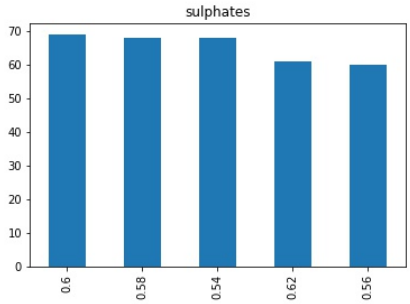
(b)


pH
50
40
30
20
10
0
3.3
3.36
3.26
3.38
3.39

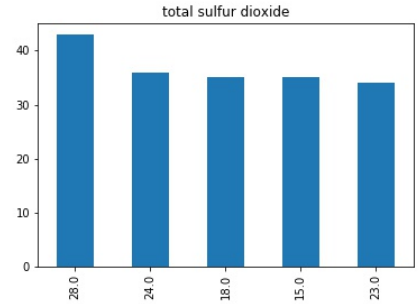
(a)


residual sugar
160
140
120
100
80
60
40
20
0
2.0
2.2
1.8
2.1
1.9

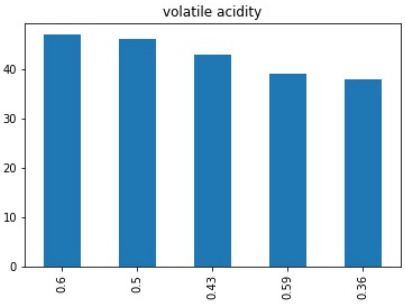
(b)


sulphates
70
60
50
40
30
20
10
0
0.6
0.58
0.54
0.62
0.56

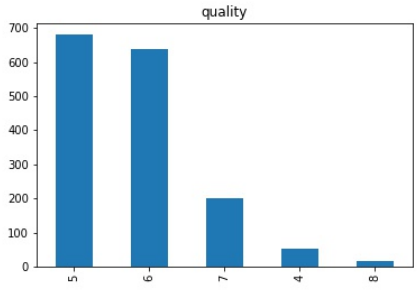
(a)


total sulfur dioxide
40
30
20
10
0
28.0
24.0
18.0
15.0
23.0

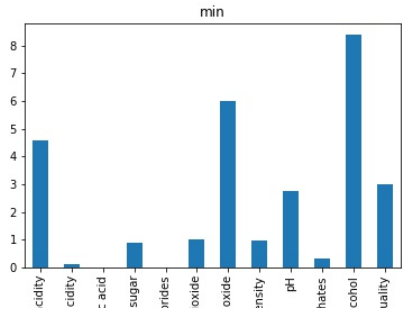
(b)


volatile acidity
0.6
0.5
0.43
0.59
0.36

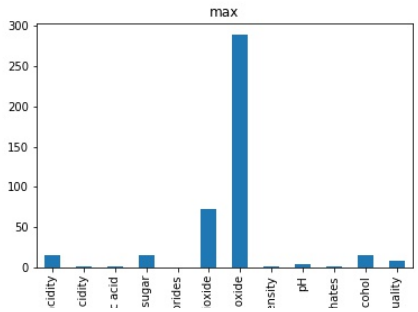
(a)


quality
5
6
7
4
8

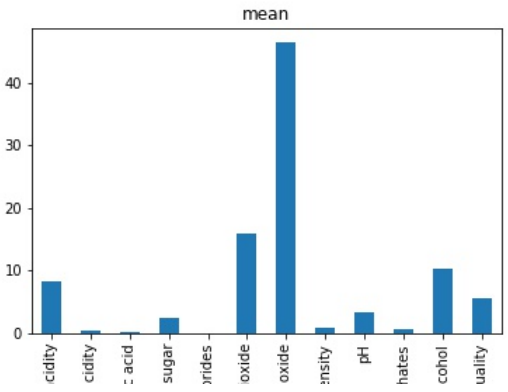
(b)


min
pH

(a)


max

(b)


mean
pH