FeatureEngineering

March 30, 2021

1 Feature engineering

1.1 Milestone 2

1.1.1 Dominik Pawlak, Przemysław Olender

[79]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
import category_encoders as ce
import warnings
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from statistics import stdev
import xgboost as xgb
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

[80]:
```
grades_df = pd.read_csv('school_grades_dataset.csv')

print(grades_df.shape)
grades_df.head()
```

```
(649, 33)
```

[80]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  ...  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  ...
1     GP   F   17       U     GT3       T     1     1  at_home     other  ...
2     GP   F   15       U     LE3       T     1     1  at_home     other  ...
3     GP   F   15       U     GT3       T     4     2   health  services  ...
4     GP   F   16       U     GT3       T     3     3    other     other  ...
```

```
   famrel  freetime  goout  Dalc  Walc  health  absences  G1  G2  G3
0       4         3      4     1     1       3         4   0  11  11
1       5         3      3     1     1       3         2   9  11  11
2       4         3      2     2     3       3         6  12  13  12
3       3         2      2     1     1       5         0  14  14  14
4       4         3      2     1     2       5         0  11  13  13

[5 rows x 33 columns]
```

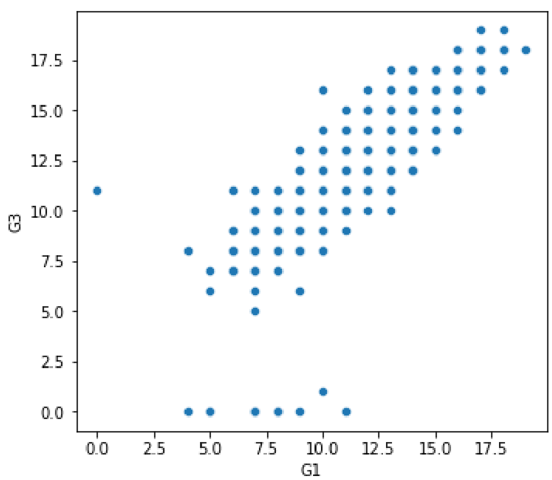
1.2 Regresja liniowa dla G1 i G2

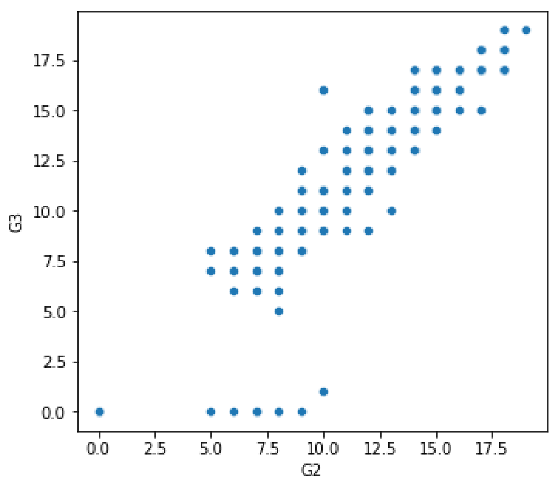
[81]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (12, 5))

sns.scatterplot(data = grades_df, x = 'G1', y = 'G3', ax = ax1)
sns.scatterplot(data = grades_df, x = 'G2', y = 'G3', ax = ax2)

plt.show()

# liniowa zalezcnosc miedzy G1, G2, i G3
```





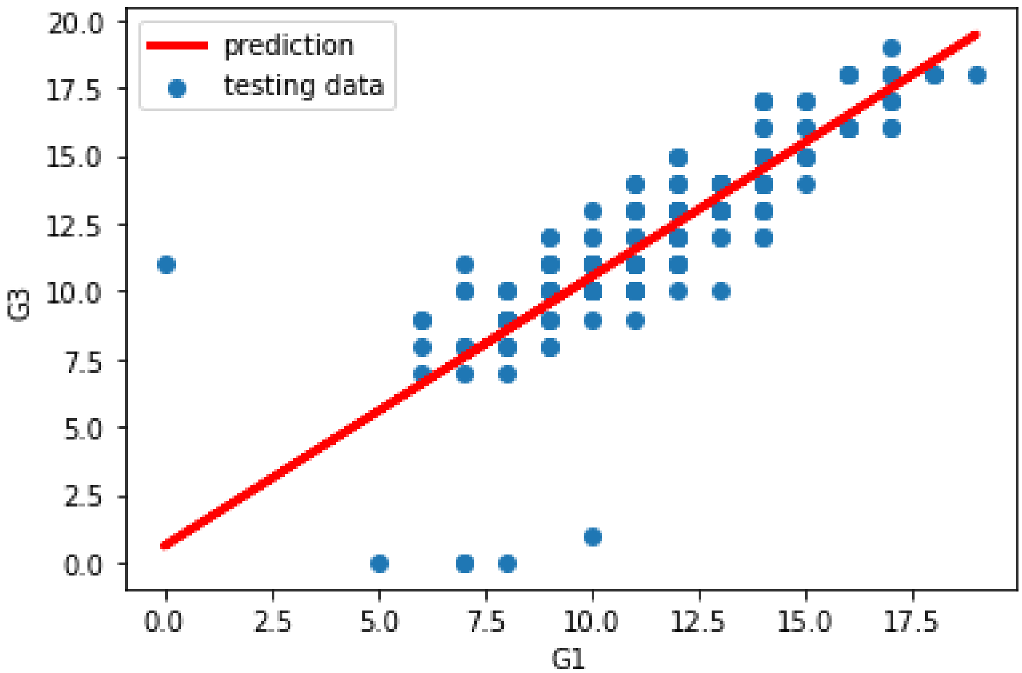
Sprawdźmy jak sprawdzi się prsoty model regresji liniowej dla G1, G2 oraz G1 i G2 jednocześnie.

[82]:
```
X1 = grades_df[['G1']]
Y = grades_df['G3']

model1 = LinearRegression()
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y, test_size = 0.3,␣
↪random_state = 1)
model1.fit(X1_train, Y1_train)
```

```python
Y1_test_predicted = model1.predict(X1_test)
```

[83]:
```python
plt.scatter(X1_test, Y1_test, label = 'testing data')
plt.plot(X1_test, Y1_test_predicted, label = 'prediction', linewidth = 3, color␣
 ↪= 'red')
plt.xlabel('G1')
plt.ylabel('G3')
plt.legend(loc = 'upper left')
plt.show()
```

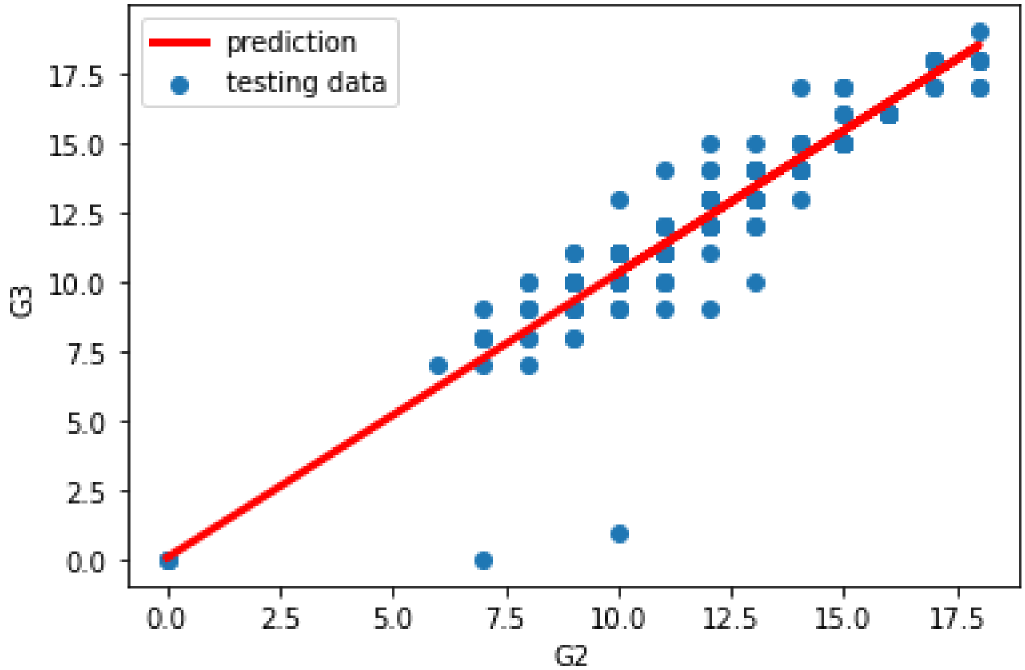


[84]:
```python
print(f'RMSE: {mean_squared_error(Y1_test, Y1_test_predicted, squared =␣
 ↪False)}')
print(f'R-squared: {model1.score(X1_test, Y1_test)}')
```

```
RMSE: 1.8966454815676594
R-squared: 0.673962037596874
```

[85]:
```python
X2 = grades_df[['G2']]
model2 = LinearRegression()
X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2, Y, test_size = 0.3,␣
 ↪random_state = 1)
model2.fit(X2_train, Y2_train)
Y2_test_predicted = model2.predict(X2_test)
```

[86]:
```
plt.scatter(X2_test, Y2_test, label = 'testing data')
plt.plot(X2_test, Y2_test_predicted, label = 'prediction', linewidth = 3, color␣
↪= 'red')
plt.xlabel('G2')
plt.ylabel('G3')
plt.legend(loc = 'upper left')
plt.show()
```



[87]:
```
print(f'RMSE: {mean_squared_error(Y2_test, Y2_test_predicted, squared =␣
↪False)}')
print(f'R-squared: {model2.score(X2_test, Y2_test)}')
```

```
RMSE: 1.23850271285229
R-squared: 0.8609760021107454
```

[88]:
```
X3 = grades_df[['G1', 'G2']]
model3 = LinearRegression()
X3_train, X3_test, Y3_train, Y3_test = train_test_split(X3, Y, test_size = 0.3,␣
↪random_state = 1)
model3.fit(X3_train, Y3_train)
Y3_test_predicted = model3.predict(X3_test)
```

[89]:
```
print(f'RMSE: {mean_squared_error(Y3_test, Y3_test_predicted, squared =␣
↪False)}')
```

```
print(f'R-squared: {model3.score(X3_test, Y3_test)}')
```

```
RMSE: 1.2238309021485727
R-squared: 0.8642503623493104
```

Najlepeij wypadła regresja oparta na G1 i G2, model oparty tylko na G2 jest minimalnie gorszy, najsłabeij prezentuje się model oparty na G1.

1.3 Przewidywanie G3 za pomocą innych zmiennych oraz bez G1 i G2

[90]:
```
grades_df[grades_df['G3'] == 0]
```

[90]:
```
    school sex  age address famsize Pstatus  Medu  Fedu      Mjob      Fjob  \
163     GP   M   18       U     LE3       T     1     1     other     other
440     MS   M   16       U     GT3       T     1     1   at_home  services
519     MS   M   16       R     GT3       T     2     1     other  services
563     MS   M   17       U     GT3       T     2     2     other     other
567     MS   M   18       R     GT3       T     3     2  services     other
583     MS   F   18       R     GT3       T     2     2     other     other
586     MS   F   17       U     GT3       T     4     2   teacher  services
597     MS   F   18       R     GT3       T     2     2   at_home     other
603     MS   F   18       R     LE3       A     4     2   teacher     other
605     MS   F   19       U     GT3       T     1     1   at_home  services
610     MS   F   19       R     GT3       A     1     1   at_home   at_home
626     MS   F   18       R     GT3       T     4     4     other   teacher
637     MS   M   18       R     GT3       T     2     1     other     other
639     MS   M   19       R     GT3       T     1     1     other  services
640     MS   M   18       R     GT3       T     4     2     other     other

     … famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
163  …      2        3      5     2     5      4        0  11   9   0
440  …      5        4      5     4     5      3        0   7   0   0
519  …      5        2      1     1     1      2        0   8   7   0
563  …      1        2      1     2     3      5        0   7   0   0
567  …      2        3      1     2     2      5        0   4   0   0
583  …      5        5      5     1     1      3        0   8   6   0
586  …      5        5      5     1     3      5        0   8   8   0
597  …      4        3      3     1     1      4        0   9   0   0
603  …      5        3      1     1     1      5        0   5   0   0
605  …      5        5      5     2     3      2        0   5   0   0
610  …      3        5      4     1     4      1        0   8   0   0
626  …      3        2      2     4     2      5        0   7   5   0
637  …      4        4      3     1     3      5        0   7   7   0
639  …      4        3      2     1     3      5        0   5   8   0
640  …      5        4      3     4     3      3        0   7   7   0

[15 rows x 33 columns]
```

Ciężko jest nie zdobyć żadnego punktu na egazminie, zakładamy, że te osoby nie podeszły i nie można przewidzieć ilości punktów, usuwamy rekordy z ramki.

[91]:
```
grades_df = grades_df[grades_df['G3'] != 0]

grades_df.shape
```

[91]:
```
(634, 33)
```

W naszej ramce mamy też dużo zmienncyh kategorycznych, zakodujmy je, żeby usprawnić działanie modeli.

[92]:
```
cat_cols = ['school', 'sex', 'address', 'famsize', 'Mjob', 'Fjob', 'reason',␣
↪'guardian', 'Pstatus', 'sex', 'school']
bin_cols = ['famsup', 'activities', 'nursery', 'internet', 'romantic',␣
↪'higher', 'paid', 'schoolsup']

grades_df_new = grades_df.drop(columns = cat_cols)
grades_df_new = grades_df.drop(columns = bin_cols)

for i in cat_cols:
    means = grades_df.groupby(i)['G3'].mean()
    grades_df_new[i] = grades_df[i].map(means)

for i in bin_cols:
    encoder = ce.OrdinalEncoder(mapping = [{'col': i, 'mapping': {'yes': 1,␣
↪'no': 0}},])
    grades_df_new[i] = encoder.fit_transform(grades_df)[i]
```

Pogrupujmy rezultat G3 w przedział, aby nie trzeba było przewidywać dokłądniej liczby - zadanie będzie prostsze.

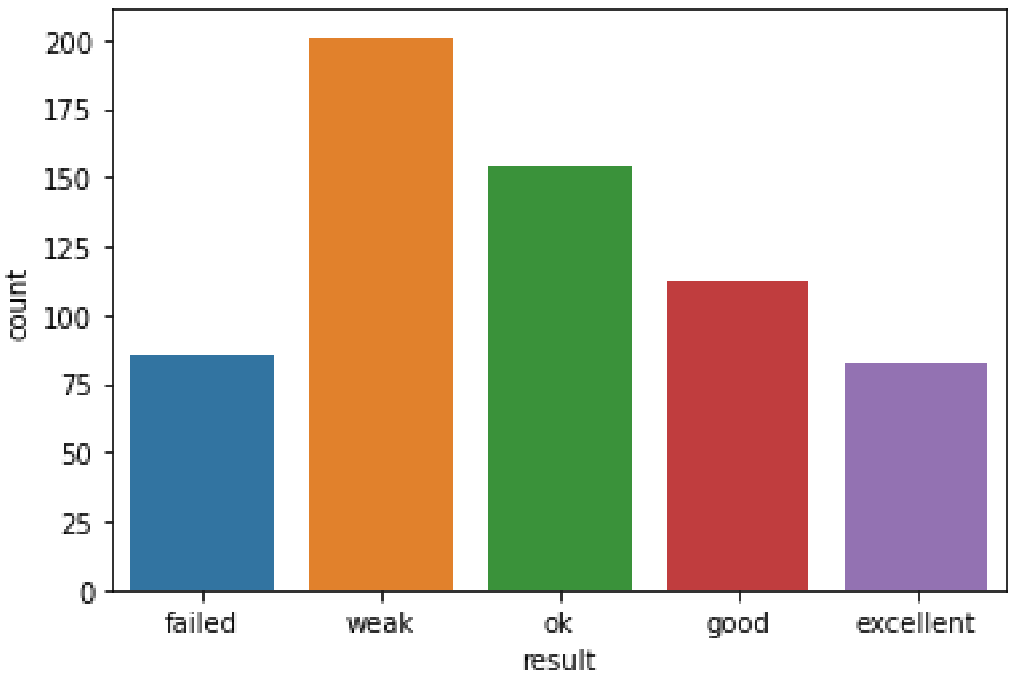
[93]:
```
names = ['failed', 'weak', 'ok', 'good', 'excellent']

grades_df_new['result'] = pd.cut(grades_df['G3'], bins=[-1, 9, 11, 13, 15, 21],␣
↪labels = names)

sns.countplot(grades_df_new['result'], order = names)
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[93]:
```
<AxesSubplot:xlabel='result', ylabel='count'>
```



[94]:

```
# funkcja liczby accuracy dla 4 modeli i za pomocą xgboost wybiera␣
 ↪najważniejsze cechy

def modelScores(X, X_train, X_test, y_train, y_test):

    importance = pd.DataFrame()
    importance['col'] = X.columns
    importance['xgb'] = 0

    model_xgb = xgb.XGBClassifier(
                    max_depth=4
                    ,learning_rate=0.2
                    ,reg_lambda=1
                    ,n_estimators=150
                    ,subsample = 0.9
                    ,colsample_bytree = 0.9)
    model_xgb.fit(X_train, y_train)

    importance['xgb'] = importance['xgb'] + model_xgb.feature_importances_ / 100
    importance = importance.sort_values(axis=0, ascending=False, by='xgb')

    print(importance.reset_index().drop('index', axis = 1))

    print(f'xgboost accuarcy: {model_xgb.score(X_test,y_test)}')
```

```
    modelLR = LogisticRegression(random_state=1, max_iter=100000)
    modelLR.fit(X_train, y_train)
    y_hat = modelLR.predict(X_test)
    print(f'Logistic Regression accuracy: {accuracy_score(y_test, y_hat)}')

    model_rf = RandomForestClassifier(n_estimators=188,
                                      max_depth=5,
                                      min_samples_split = 7,
                                      max_features = len(X_train.columns),
                                      random_state=0,
                                      n_jobs = 15)
    model_rf.fit(X_train, y_train)
    print(f'RandomForestClassifier accuracy: {model_rf.score(X_test,y_test)}')

    model1 = DecisionTreeClassifier(random_state=1)
    clf = BaggingClassifier(base_estimator=model_rf,
                            n_estimators=100, random_state=0)
    clf.fit(X_train, y_train)
    print(f'DecisionTreeClassifier accuracy: {clf.score(X_test,y_test)}')

    return importance['col'].head(10)
```

[95]:
```
X = grades_df_new.drop(['G3', 'result'], axis = 1)
Y = grades_df_new['result']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
 ↪random_state=42)
```

[96]:
```
cols = modelScores(X, X_train, X_test, y_train, y_test)
```

	col	xgb
0	G2	0.002208
1	G1	0.000905
2	higher	0.000437
3	paid	0.000432
4	schoolsup	0.000401
5	school	0.000306
6	famsize	0.000249
7	Pstatus	0.000247
8	freetime	0.000245
9	Dalc	0.000244
10	Walc	0.000242
11	Fjob	0.000241
12	Mjob	0.000236
13	health	0.000234
14	traveltime	0.000226
15	absences	0.000225

```
16          age  0.000218
17     failures  0.000217
18       reason  0.000216
19    studytime  0.000214
20     internet  0.000207
21         Fedu  0.000197
22   activities  0.000194
23      nursery  0.000194
24        goout  0.000181
25         Medu  0.000180
26          sex  0.000174
27     guardian  0.000159
28       famrel  0.000155
29       famsup  0.000146
30     romantic  0.000140
31      address  0.000131
xgboost accuarcy: 0.6596858638743456
Logistic Regression accuracy: 0.6492146596858639
RandomForestClassifier accuracy: 0.743455497382199
DecisionTreeClassifier accuracy: 0.7539267015706806
```

[97]:

```python
X = grades_df_new[cols]
Y = grades_df_new['result']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
↪random_state=42)

modelScores(X, X_train, X_test, y_train, y_test)
```

```
         col       xgb
0         G2  0.004274
1         G1  0.001353
2  schoolsup  0.000636
3       paid  0.000598
4     school  0.000589
5       Dalc  0.000565
6    famsize  0.000516
7     higher  0.000515
8   freetime  0.000510
9    Pstatus  0.000444
xgboost accuarcy: 0.6910994764397905
Logistic Regression accuracy: 0.6910994764397905
RandomForestClassifier accuracy: 0.7382198952879581
DecisionTreeClassifier accuracy: 0.743455497382199
```

[97]:

```
0            G2
1            G1
```

```
4     schoolsup
3          paid
5        school
9          Dalc
6       famsize
2        higher
8      freetime
7       Pstatus
Name: col, dtype: object
```

[98]:
```python
X = grades_df_new.drop(['G3', 'result'], axis = 1)
Y = grades_df_new['G3']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
 ↪random_state=42)
```

[99]:
```python
cols = modelScores(X, X_train, X_test, y_train, y_test)
```

	col	xgb
0	G2	0.001785
1	G1	0.000653
2	paid	0.000508
3	Pstatus	0.000445
4	higher	0.000393
5	Fedu	0.000322
6	failures	0.000314
7	school	0.000289
8	guardian	0.000284
9	Dalc	0.000273
10	schoolsup	0.000270
11	Mjob	0.000253
12	age	0.000250
13	famsize	0.000248
14	freetime	0.000240
15	Fjob	0.000240
16	goout	0.000238
17	studytime	0.000233
18	internet	0.000231
19	famrel	0.000224
20	traveltime	0.000222
21	Walc	0.000221
22	address	0.000220
23	health	0.000207
24	absences	0.000205
25	reason	0.000199
26	Medu	0.000199
27	romantic	0.000192

```
28         sex  0.000176
29  activities  0.000157
30     nursery  0.000156
31      famsup  0.000151
xgboost accuarcy: 0.3612565445026178
Logistic Regression accuracy: 0.3089005235602094
RandomForestClassifier accuracy: 0.450261780104712
DecisionTreeClassifier accuracy: 0.45549738219895286
```

[100]:
```
X = grades_df_new[cols]
Y = grades_df_new['G3']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
↪random_state=42)

modelScores(X, X_train, X_test, y_train, y_test)
```

```
        col       xgb
0        G2  0.003245
1      paid  0.001138
2        G1  0.001052
3   Pstatus  0.000766
4    higher  0.000672
5  failures  0.000659
6  guardian  0.000653
7      Dalc  0.000627
8      Fedu  0.000596
9    school  0.000592
xgboost accuarcy: 0.42408376963350786
Logistic Regression accuracy: 0.36649214659685864
RandomForestClassifier accuracy: 0.387434554973822
DecisionTreeClassifier accuracy: 0.4397905759162304
```

[100]:
```
0          G2
2        paid
1          G1
3     Pstatus
4      higher
6    failures
8    guardian
9        Dalc
5        Fedu
7      school
Name: col, dtype: object
```

Wyniki gorsze niż dla prostej regresji liniowej.

Stworzylismy prosty model opierający sięn a G1 i G2, spróbujmy teraz bez tych atrybutów.

[101]:
```
grades_df_new = grades_df_new.drop(columns = ['G1', 'G2'])
```

Przewidywanie przedziału oceny za pomocą wszystkich kolumn:

[102]:
```
X = grades_df_new.drop(['G3', 'result'], axis = 1)
Y = grades_df_new['result']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
↪random_state=42)
```

[103]:
```
cols = modelScores(X, X_train, X_test, y_train, y_test)
```

```
           col       xgb
0     failures  0.000789
1       higher  0.000613
2       school  0.000549
3    schoolsup  0.000470
4    studytime  0.000403
5         paid  0.000360
6     internet  0.000339
7      Pstatus  0.000336
8     freetime  0.000331
9         Dalc  0.000326
10    absences  0.000324
11        Walc  0.000319
12  traveltime  0.000318
13     nursery  0.000315
14         sex  0.000313
15        Fedu  0.000312
16        Mjob  0.000303
17        Medu  0.000291
18      reason  0.000277
19       goout  0.000268
20         age  0.000262
21      famsup  0.000260
22        Fjob  0.000252
23    guardian  0.000250
24      famrel  0.000247
25      health  0.000244
26    romantic  0.000243
27     address  0.000232
28  activities  0.000232
29     famsize  0.000222
xgboost accuarcy: 0.3193717277486911
Logistic Regression accuracy: 0.28272251308900526
RandomForestClassifier accuracy: 0.31413612565445026
DecisionTreeClassifier accuracy: 0.32460732984293195
```

Przewidywanie przedziału oceny za pomocą najważnijeszych kolumn:

[104]:
```
X = grades_df_new[cols]
Y = grades_df_new['result']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
↪random_state=42)
modelScores(X, X_train, X_test, y_train, y_test)
```

```
         col       xgb
0   failures  0.001669
1     higher  0.001527
2     school  0.001031
3  schoolsup  0.000964
4       paid  0.000941
5  studytime  0.000835
6   freetime  0.000795
7   internet  0.000780
8       Dalc  0.000775
9    Pstatus  0.000683
xgboost accuarcy: 0.2879581151832461
Logistic Regression accuracy: 0.2931937172774869
RandomForestClassifier accuracy: 0.28272251308900526
DecisionTreeClassifier accuracy: 0.27225130890052357
```

[104]:
```
0     failures
1       higher
2       school
3    schoolsup
5         paid
4    studytime
8     freetime
6     internet
9         Dalc
7      Pstatus
Name: col, dtype: object
```

Przewidywanie dokładnej oceny za pomocą wszystkich kolumn:

[105]:
```
X = grades_df_new.drop(['G3', 'result'], axis = 1)
Y = grades_df_new['G3']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
↪random_state=42)

cols = modelScores(X, X_train, X_test, y_train, y_test)
```

```
          col       xgb
```

```
0     failures  0.000595
1       school  0.000496
2       higher  0.000474
3      Pstatus  0.000407
4         paid  0.000399
5    schoolsup  0.000396
6     guardian  0.000391
7    studytime  0.000359
8         Dalc  0.000357
9         Fedu  0.000356
10    absences  0.000347
11      reason  0.000333
12        Mjob  0.000323
13        Fjob  0.000321
14         age  0.000318
15     address  0.000303
16    internet  0.000301
17    freetime  0.000295
18       goout  0.000294
19  traveltime  0.000293
20         sex  0.000279
21        Medu  0.000275
22      health  0.000274
23  activities  0.000271
24      famsup  0.000270
25     famsize  0.000270
26      famrel  0.000259
27     nursery  0.000259
28        Walc  0.000248
29    romantic  0.000235
xgboost accuarcy: 0.18848167539267016
Logistic Regression accuracy: 0.17277486910994763
RandomForestClassifier accuracy: 0.17277486910994763
DecisionTreeClassifier accuracy: 0.18848167539267016
```

Przewidywanie dokładnej oceny za pomocą najważnieszych kolumn:

[106]:
```
X = grades_df_new[cols]
Y = grades_df_new['G3']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,␣
 ↪random_state=42)

modelScores(X, X_train, X_test, y_train, y_test)
```

```
        col       xgb
0  failures  0.001404
1    higher  0.001228
```

```
2       paid  0.001070
3  schoolsup  0.001049
4     school  0.000984
5   guardian  0.000903
6    Pstatus  0.000872
7       Dalc  0.000855
8       Fedu  0.000836
9  studytime  0.000799
xgboost accuarcy: 0.1099476439790576
Logistic Regression accuracy: 0.17277486910994763
RandomForestClassifier accuracy: 0.13612565445026178
DecisionTreeClassifier accuracy: 0.16230366492146597
```

[106]:
```
0     failures
2       higher
4         paid
5    schoolsup
1       school
6     guardian
3      Pstatus
8         Dalc
9         Fedu
7    studytime
Name: col, dtype: object
```