Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4

Вариант №15

Выполнил:
студент группы ИУ5-63
Миронова Александра
Подпись и дата:
30.05.22

Проверил:
Юрий Евгеньевич Гапанюк
Подпись и дата:

Москва, 2022 г.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите следующие модели:
    - одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
    - SVM;
    - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

Результат:

Lab_4.md

Загрузка и первичный анализ данных

Для выполнения задания был выбран датасет библиотеки sklearn с данными о характеристиках вин и их рейтингом цен.

https://scikit-learn.org/stable/modules/classes.html?highlight=sklearn%20datasets#module-sklearn.datasets

```
from operator import itemgetter
import pydotplus
import graphviz
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, plot_confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.svm import SVC
from IPython.display import Image
from io import StringIO
import numpy as np
import pandas as pd
import seaborn as sns
import math
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from sklearn.datasets import *
%matplotlib inline
sns.set(style="ticks")
```

```
ds = load_wine()
data = pd.DataFrame(data= np.c_[ds['data'], ds['target']],
                    columns= list(ds['feature_names']) + ['class'])
data.dtypes
```

```
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
```

```
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
class                           float64
dtype: object
```

```
data.shape
```

```
(178, 14)
```

```
data
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonfla
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39
...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56

178 rows × 14 columns

```
np.unique(data['class'])
```

```
array([0., 1., 2.])
```

```
np.where(pd.isnull(data))
```

```
(array([], dtype=int64), array([], dtype=int64))
```

В датасете нет пропусков

Будем решать задачу классификации. Поскольку в датасете имеется большое количество столблов с небинарными данными, то мы будем использовать сетрики для оценки моделей с небинарными данными. Для оценки обученных моделей планируется использование метрики accuracy, Confusion matrix. Предварительно проверим, можно ли применить их.

```
np.unique(data['class'], return_counts=True)
```

```
(array([0., 1., 2.]), array([59, 71, 48], dtype=int64))
```

Классы сбалансированы. Можно использовать метрику accuracy.

Корелляционная матрица

```
data.corr()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	to

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	to
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.
class	-0.328222	0.437776	-0.049643	0.517859	-0.209179	-0

Разделение выборки на обучающую и тестовую.

```
y=np.array(data["class"])
X=np.array(data.drop(["class"], axis=1))
X, y
```

```
(array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
         1.065e+03],
        [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
         1.050e+03],
        [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
         1.185e+03],
        ...,
        [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
         8.350e+02],
        [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
         8.400e+02],
        [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
         5.600e+02]]),
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2., 2., 2., 2.,
       2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2.,
       2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2.,
       2., 2., 2., 2., 2., 2., 2., 2.]))
```

Разделение выборки на обучающую и тестовую

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,shuffle=True)
```

Выборка разделена на обучающую X_train, y_train и тестовую X_test, y_test

Обучение моделей

LogisticRegression

```
LRmodel = LogisticRegression(max_iter=10000, random_state=0)
```

```
LRmodel.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=10000, random_state=0)
```

```
predLR_y_test = LRmodel.predict(X_test)
predLR_y_test
```

```
array([2., 1., 0., 1., 0., 2., 1., 0., 2., 1., 0., 0., 1., 0., 1., 1., 2.,
       0., 1., 0., 0., 1., 1., 0., 0., 2., 0., 0., 0., 2., 1., 2., 2., 0.,
       1., 1., 1., 1., 1., 0., 0., 1., 2., 0., 0., 0., 1., 0., 0., 0., 1.,
       2., 2., 0.])
```

Оценка качества

Accuracy

```
accuracy_score(y_test, predLR_y_test)
```

```
0.9629629629629629
```

Confusion matrix

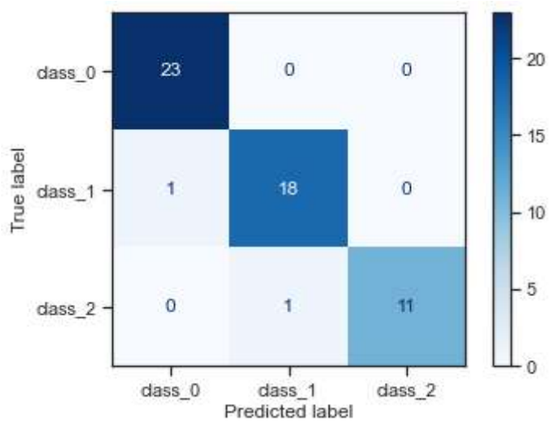
```
# Для небинарной классификации
confusion_matrix(y_test, predLR_y_test, labels=[0, 1, 2])
```

```
array([[23,  0,  0],
       [ 1, 18,  0],
       [ 0,  1, 11]], dtype=int64)
```

```
plot_confusion_matrix(LRmodel, X_test, y_test,
                      display_labels=ds.target_names, cmap=plt.cm.Blues)
```

```
C:\Users\Alexandra\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
  warnings.warn(msg, category=FutureWarning)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x21ed5928e50>
```



SVM

```
SVCmodel = SVC(kernel='linear', C=2, gamma="auto", decision_function_shape='ovo')
SVCmodel.fit(X_train, y_train)
```

```
SVC(C=2, decision_function_shape='ovo', gamma='auto', kernel='linear')
```

```
predSVC_y_test = SVCmodel.predict(X_test)
predSVC_y_test
```

```
array([2., 1., 0., 1., 0., 2., 1., 0., 2., 1., 0., 0., 1., 0., 1., 1., 2.,
       0., 1., 0., 0., 1., 1., 0., 0., 2., 0., 0., 0., 2., 1., 2., 2., 0.,
       1., 1., 1., 1., 1., 0., 0., 1., 2., 0., 0., 0., 1., 0., 0., 0., 1.,
       2., 2., 0.])
```

```
y_test
```

```
array([2., 1., 0., 1., 0., 2., 1., 0., 2., 1., 0., 0., 1., 0., 1., 1., 2.,
       0., 1., 0., 0., 1., 2., 1., 0., 2., 0., 0., 0., 2., 1., 2., 2., 0.,
       1., 1., 1., 1., 1., 0., 0., 1., 2., 0., 0., 0., 1., 0., 0., 0., 1.,
       2., 2., 0.])
```

Оценка качества

Accuracy

```
accuracy_score(y_test, predSVC_y_test)
```

```
0.9629629629629629
```

Confusion matrix

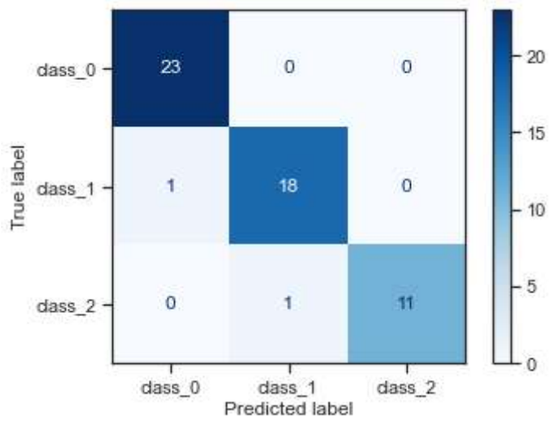
```
confusion_matrix(y_test, predSVC_y_test, labels=[0, 1, 2])
```

```
array([[23,  0,  0],
       [ 1, 18,  0],
       [ 0,  1, 11]], dtype=int64)
```

```
plot_confusion_matrix(SVCmodel, X_test, y_test,
                      display_labels=ds.target_names, cmap=plt.cm.Blues)
```

```
C:\Users\Alexandra\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
  warnings.warn(msg, category=FutureWarning)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x21ee6eff0a0>
```



Дерево решений

```
DecisionTreeModel = DecisionTreeClassifier()
DecisionTreeModel.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
predTree_y_test = DecisionTreeModel.predict(X_test)
predTree_y_test
```

```
array([2., 1., 0., 1., 0., 2., 1., 0., 2., 1., 0., 1., 1., 0., 1., 1., 2.,
       0., 1., 0., 0., 1., 2., 1., 0., 2., 0., 0., 0., 2., 1., 2., 2., 0.,
       1., 1., 1., 1., 1., 0., 0., 2., 2., 0., 0., 0., 1., 0., 0., 0., 1.,
       2., 2., 0.])
```

Оценка качества

Accuracy

```
accuracy_score(y_test, predTree_y_test)
```

```
0.9259259259259259
```

Confusion matrix

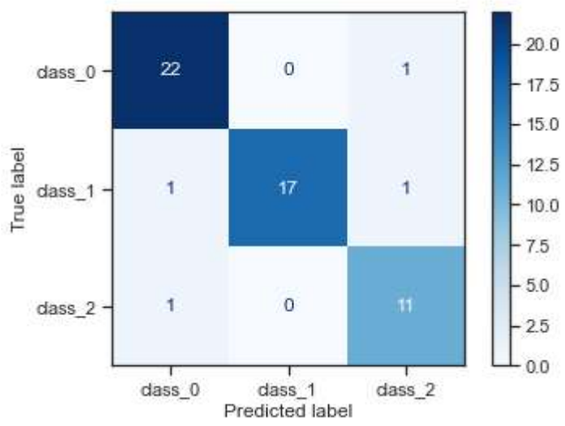
```
confusion_matrix(y_test, predTree_y_test, labels=[0, 1, 2])
```

```
array([[22,  0,  1],
       [ 1, 17,  1],
       [ 1,  0, 11]], dtype=int64)
```

```
plot_confusion_matrix(DecisionTreeModel, X_test, y_test,
                      display_labels=ds.target_names, cmap=plt.cm.Blues)
```

```
C:\Users\Alexandra\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\deprecation.py:87:
  warnings.warn(msg, category=FutureWarning)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x21ee907cbe0>
```



Общие результаты оценок

```
results_metrics = dict()
model_list = ['logistic_regression', 'svc','tree']
```

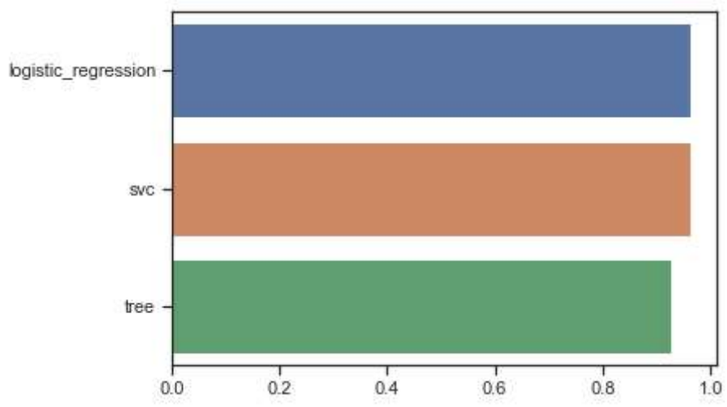
```
results_metrics['accuracy'] = [accuracy_score(y_test, predLR_y_test), accuracy_score(y_test, predSVC_y_test),
```

```
results_metrics['confusion_matrix'] = [2, 2, 2]
```

```
sns.barplot(results_metrics['accuracy'], model_list)
```

```
C:\Users\Alexandra\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_decorators.py:36: Futur
  warnings.warn(
```

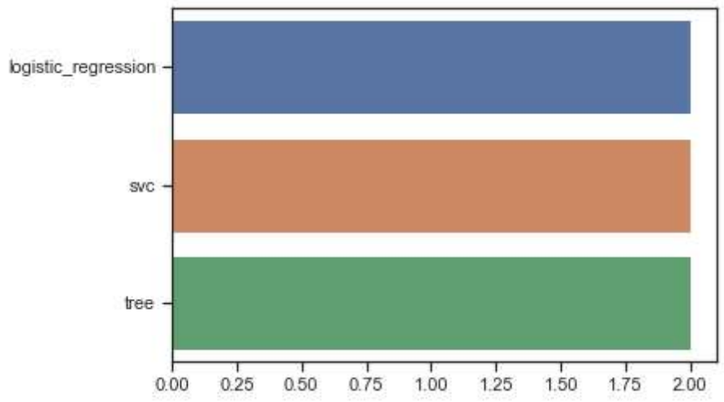
```
<AxesSubplot:>
```




```
sns.barplot(results_metrics['confusion_matrix'], model_list)
```

```
C:\Users\Alexandra\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_decorators.py:36: Futur
  warnings.warn(
```

```
<AxesSubplot:>
```



Оценки моделей в каждой метрике имеют соответственно равные значения

Важность признаков в дереве решений

Дерево решений в png

```
def get_png_tree(tree_model_param, feature_names_param):
    dot_data = StringIO()
    export_graphviz(tree_model_param, out_file=dot_data, feature_names=feature_names_param,
                    filled=True, rounded=True, special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph.create_png()
```

```
X_train
```

```
array([[1.208e+01, 1.830e+00, 2.320e+00, ..., 1.080e+00, 2.270e+00,
        4.800e+02],
       [1.247e+01, 1.520e+00, 2.200e+00, ..., 1.160e+00, 2.630e+00,
        9.370e+02],
       [1.387e+01, 1.900e+00, 2.800e+00, ..., 1.250e+00, 3.400e+00,
        9.150e+02],
       ...,
       [1.349e+01, 1.660e+00, 2.240e+00, ..., 9.800e-01, 2.780e+00,
        4.720e+02],
       [1.293e+01, 2.810e+00, 2.700e+00, ..., 7.700e-01, 2.310e+00,
        6.000e+02],
       [1.305e+01, 1.650e+00, 2.550e+00, ..., 1.120e+00, 2.510e+00,
        1.105e+03]])
```

```
data.columns
```

```
Index(['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
       'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
       'proanthocyanins', 'color_intensity', 'hue',
       'od280/od315_of_diluted_wines', 'proline', 'class'],
      dtype='object')
```

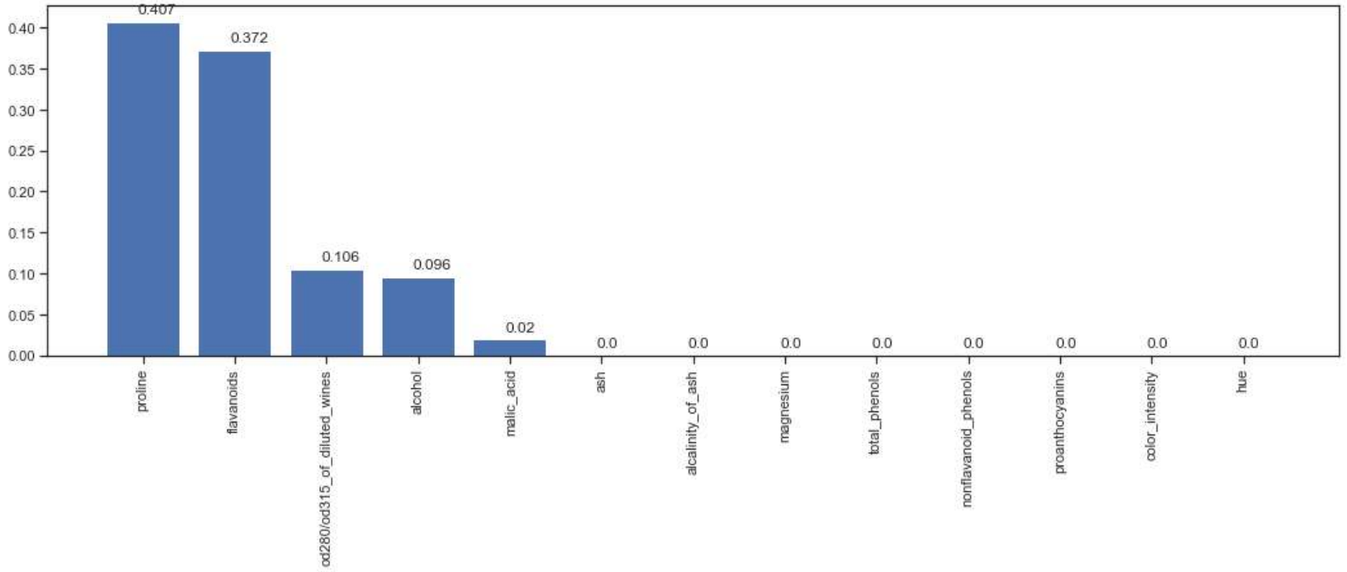
```
#Image(get_png_tree(tree, df_X_train.columns), height='100%')
with open('./tree.png', 'wb') as f:
    f.write(get_png_tree(DecisionTreeModel,  data.drop(["class"], axis=1).columns))
```

```
---------------------------------------------------------------------------

InvocationException                       Traceback (most recent call last)

C:\Users\ALEXAN~1\AppData\Local\Temp/ipykernel_4984/1331949745.py in <module>
      1 #Image(get_png_tree(tree, df_X_train.columns), height='100%')
      2 with open('./tree.png', 'wb') as f:
----> 3     f.write(get_png_tree(DecisionTreeModel,  data.drop(["class"], axis=1).columns))


C:\Users\ALEXAN~1\AppData\Local\Temp/ipykernel_4984/1399891933.py in get_png_tree(tree_model_param, feature_n
      4                     filled=True, rounded=True, special_characters=True)
      5     graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
----> 6     return graph.create_png()


~\AppData\Local\Programs\Python\Python310\lib\site-packages\pydotplus\graphviz.py in <lambda>(f, prog)
   1795             self.__setattr__(
   1796                 'create_' + frmt,
-> 1797                 lambda f=frmt, prog=self.prog: self.create(format=f, prog=prog)
   1798             )
   1799             f = self.__dict__['create_' + frmt]


~\AppData\Local\Programs\Python\Python310\lib\site-packages\pydotplus\graphviz.py in create(self, prog, forma
   1957             self.progs = find_graphviz()
   1958             if self.progs is None:
-> 1959                 raise InvocationException(
   1960                     'GraphViz\'s executables not found')
   1961

InvocationException: GraphViz's executables not found
```

```
Image(get_png_tree(DecisionTreeModel, X_train.columns), height='100%')
```

Правила дерева решений

```
def draw_feature_importances(tree_model, X_dataset, figsize=(18,5)):
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importances_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
    # Названия признаков
    labels = [x for x,_ in sorted_list]
    # Важности признаков
    data = [x for _,x in sorted_list]
    # Вывод графика
    fig, ax = plt.subplots(figsize=figsize)
    ind = np.arange(len(labels))
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    # Вывод значений
    for a,b in zip(ind, data):
        plt.text(a-0.05, b+0.01, str(round(b,3)))
    plt.show()
    return labels, data


draw_feature_importances(DecisionTreeModel, data.drop(["class"], axis=1))
```




```
(['proline',
  'flavanoids',
  'od280/od315_of_diluted_wines',
  'alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
```

```
 'magnesium',
 'total_phenols',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue'],
[0.40685320507448963,
 0.37183383991894625,
 0.1055713952247566,
 0.09605902009926777,
 0.019682539682539673,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0])
```