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

«Разведочный анализ данных. Исследование и визуализация данных.»

Выполнил:
студент группы ИУ5-63Б
Воронова О. А.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2022 г.

Лабораторная работа №4

Линейные модели, SVM и деревья решений

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.

В качестве исходных данных возьмём датасет расхода топлива автомобилей в Канаде 2022 года В этой лабораторной работе будем решать задачу классификации Целевой признак - Cylinders

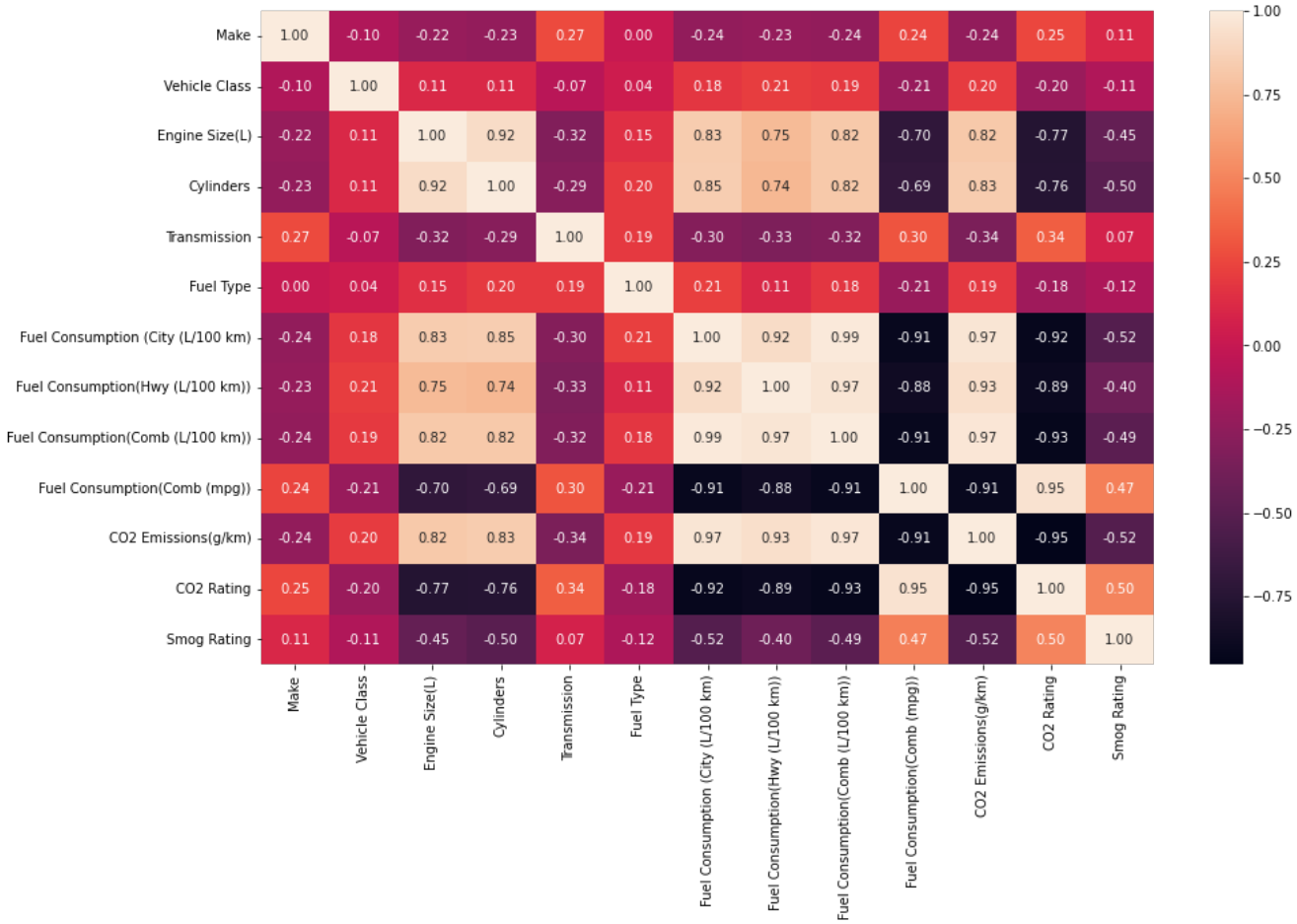
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from typing import Tuple
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import balanced_accuracy_score
from sklearn.tree import DecisionTreeClassifier
import graphviz
from sklearn.tree import export_graphviz
from sklearn import tree
from operator import itemgetter
data = pd.read_csv("Fuel.csv")
#Первые 5 записей датасета
data.head()
```

	Model Year	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fue Consumptio (Cit (L/100 km
0	2022	Acura	ILX	Compact	2.4	4	AM8	Z	9.9
1	2022	Acura	MDX SH-AWD	SUV: Small	3.5	6	AS10	Z	12.6
2	2022	Acura	RDX SH-AWD	SUV: Small	2.0	4	AS10	Z	11.0
3	2022	Acura	RDX SH-AWD A-SPEC	SUV: Small	2.0	4	AS10	Z	11.3
4	2022	Acura	TLX SH-AWD	Compact	2.0	4	AS10	Z	11.2

```
#Проверка наличия пустых значений
data.isnull().sum()
Model Year                                0
Make                                      0
Model                                     0
Vehicle Class                             0
Engine Size(L)                            0
Cylinders                                 0
Transmission                              0
Fuel Type                                 0
Fuel Consumption (City (L/100 km)         0
Fuel Consumption(Hwy (L/100 km))          0
Fuel Consumption(Comb (L/100 km))         0
Fuel Consumption(Comb (mpg))              0
CO2 Emissions(g/km)                       0
CO2 Rating                                0
Smog Rating                               0
dtype: int64
#Размер исходного датасета
data.shape
(946, 15)
#Проверка типов
data.dtypes
Model Year                                int64
Make                                     object
```

```
Model                                    object
Vehicle Class                            object
Engine Size(L)                          float64
Cylinders                                 int64
Transmission                             object
Fuel Type                                object
Fuel Consumption (City (L/100 km)       float64
Fuel Consumption(Hwy (L/100 km))        float64
Fuel Consumption(Comb (L/100 km))       float64
Fuel Consumption(Comb (mpg))              int64
CO2 Emissions(g/km)                       int64
CO2 Rating                                int64
Smog Rating                               int64
dtype: object
```

```
#Удаление ненужных столбцов
data = data.drop(columns=["Model", "Model Year"], axis=1)
#Кодирование категориальных признаков
LE = LabelEncoder()
for column in ["Fuel Type","Make", "Vehicle Class",
"Transmission"]:
    data[column] = LE.fit_transform(data[column])
fig, ax = plt.subplots(figsize=(15,9))
sns.heatmap(data.corr(method='pearson'), ax=ax, annot=True,
fmt='.2f')
<AxesSubplot:>
```




```
#Выделяем записи, где присутствуют 4 или 6 цилиндров
```

```
#Для бинарной классификации
data = data.loc[data["Cylinders"].isin([4,6])]
data.shape
(699, 13)
xArray = data.drop("Cylinders", axis=1)
yArray = data["Cylinders"]
#Разделяем выборку для обучения модели
trainX, testX, trainY, testY = train_test_split(xArray, yArray,
test_size=0.2, random_state=1)
```

Линейная регрессия

```
#Обучение модели
LR = LogisticRegression()
LR.fit(trainX, trainY)
D:\Anaconda\lib\site-
packages\sklearn\linear_model\_logistic.py:763:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data
as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
    https://scikit-
learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression()
#Предсказание целевого признака
predict = LR.predict(testX)
predict
array([6, 6, 4, 6, 6, 4, 6, 4, 4, 4, 4, 6, 6, 6, 4, 4, 6, 4, 4,
4, 4, 6,
       4, 4, 4, 6, 6, 4, 6, 4, 4, 6, 6, 4, 6, 4, 6, 6, 6, 4, 6,
6, 6, 6,
       4, 6, 4, 4, 4, 6, 4, 4, 6, 6, 4, 4, 6, 6, 6, 6, 6, 6, 6,
4, 6, 4,
       4, 6, 4, 4, 6, 6, 4, 4, 6, 6, 6, 4, 4, 4, 6, 4, 4, 6, 6,
4, 4, 4,
       4, 4, 4, 4, 6, 6, 6, 4, 6, 4, 6, 4, 6, 6, 4, 4, 6, 4, 4,
4, 6, 6,
       4, 4, 6, 6, 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 4, 4, 6,
6, 6, 4,
       4, 4, 4, 4, 6, 6, 6, 4], dtype=int64)
#Вычисление вероятности попадания в той или иной класс
predictpr = LR.predict_proba(testX)
predictpr
array([[9.89144089e-02, 9.01085591e-01],
       [1.72492351e-02, 9.82750765e-01],
       [9.79594360e-01, 2.04056401e-02],
```

[1.05608870e-02, 9.89439113e-01],
[1.46493106e-01, 8.53506894e-01],
[9.98315700e-01, 1.68429986e-03],
[3.63899455e-03, 9.96361005e-01],
[5.26785162e-01, 4.73214838e-01],
[9.96458452e-01, 3.54154839e-03],
[9.93006653e-01, 6.99334657e-03],
[9.99996677e-01, 3.32323644e-06],
[5.00806757e-02, 9.49919324e-01],
[9.63162024e-02, 9.03683798e-01],
[1.25412220e-01, 8.74587780e-01],
[9.99698106e-01, 3.01894226e-04],
[9.80460978e-01, 1.95390224e-02],
[2.47924207e-03, 9.97520758e-01],
[9.47786291e-01, 5.22137094e-02],
[9.09163301e-01, 9.08366994e-02],
[9.94459334e-01, 5.54066598e-03],
[9.95723692e-01, 4.27630766e-03],
[1.93570270e-01, 8.06429730e-01],
[9.94668161e-01, 5.33183935e-03],
[9.92621630e-01, 7.37837005e-03],
[9.94904483e-01, 5.09551669e-03],
[8.00859072e-03, 9.91991409e-01],
[4.33876257e-03, 9.95661237e-01],
[9.99993865e-01, 6.13475292e-06],
[1.09939318e-01, 8.90060682e-01],
[9.89697772e-01, 1.03022279e-02],
[9.94619506e-01, 5.38049418e-03],
[4.11690510e-02, 9.58830949e-01],
[9.96279690e-02, 9.00372031e-01],
[9.97175642e-01, 2.82435762e-03],
[6.53860167e-02, 9.34613983e-01],
[9.99984038e-01, 1.59616419e-05],
[9.83623387e-02, 9.01637661e-01],
[3.52376287e-03, 9.96476237e-01],
[4.24047760e-02, 9.57595224e-01],
[9.96348000e-01, 3.65199999e-03],
[4.29495914e-04, 9.99570504e-01],
[6.52538183e-02, 9.34746182e-01],
[2.41874104e-03, 9.97581259e-01],
[5.95575386e-02, 9.40442461e-01],
[9.81371543e-01, 1.86284570e-02],
[4.39563877e-01, 5.60436123e-01],
[9.94907100e-01, 5.09290048e-03],
[9.99434368e-01, 5.65631912e-04],
[9.81371543e-01, 1.86284570e-02],
[5.55434266e-02, 9.44456573e-01],
[9.99987071e-01, 1.29294613e-05],
[9.85710941e-01, 1.42890592e-02],
[7.03803891e-02, 9.29619611e-01],
[6.40231378e-03, 9.93597686e-01],

[9.99873498e-01, 1.26501882e-04],
[9.99970235e-01, 2.97651752e-05],
[4.20469102e-01, 5.79530898e-01],
[3.29922404e-01, 6.70077596e-01],
[8.36228760e-02, 9.16377124e-01],
[7.26224725e-02, 9.27377528e-01],
[1.61149963e-01, 8.38850037e-01],
[1.90276531e-02, 9.80972347e-01],
[1.59959762e-01, 8.40040238e-01],
[9.99955813e-01, 4.41869588e-05],
[6.04428505e-02, 9.39557149e-01],
[9.58202416e-01, 4.17975843e-02],
[9.95501174e-01, 4.49882650e-03],
[2.21180558e-01, 7.78819442e-01],
[9.97096408e-01, 2.90359222e-03],
[9.81458081e-01, 1.85419195e-02],
[9.61901646e-05, 9.99903810e-01],
[1.50461790e-01, 8.49538210e-01],
[9.58149024e-01, 4.18509763e-02],
[9.95255771e-01, 4.74422899e-03],
[6.64627264e-02, 9.33537274e-01],
[3.06485489e-02, 9.69351451e-01],
[1.09940980e-01, 8.90059020e-01],
[9.99869173e-01, 1.30827221e-04],
[9.96382058e-01, 3.61794224e-03],
[9.36526615e-01, 6.34733854e-02],
[7.50516398e-02, 9.24948360e-01],
[9.97546139e-01, 2.45386083e-03],
[9.99485091e-01, 5.14909271e-04],
[3.76510076e-01, 6.23489924e-01],
[7.52490804e-03, 9.92475092e-01],
[9.51063457e-01, 4.89365426e-02],
[9.83493983e-01, 1.65060170e-02],
[9.99975833e-01, 2.41670693e-05],
[9.99244014e-01, 7.55986342e-04],
[9.92522388e-01, 7.47761214e-03],
[9.99941977e-01, 5.80228290e-05],
[9.99810317e-01, 1.89682704e-04],
[7.04604897e-02, 9.29539510e-01],
[4.80916316e-03, 9.95190837e-01],
[9.50981297e-02, 9.04901870e-01],
[9.89779997e-01, 1.02200034e-02],
[6.89193392e-02, 9.31080661e-01],
[6.05215305e-01, 3.94784695e-01],
[1.31458016e-01, 8.68541984e-01],
[9.99995684e-01, 4.31616190e-06],
[7.84822642e-05, 9.99921518e-01],
[8.63876818e-02, 9.13612318e-01],
[9.97594842e-01, 2.40515824e-03],
[9.02477994e-01, 9.75220062e-02],
[5.26549902e-02, 9.47345010e-01],

```
       [9.96275618e-01, 3.72438238e-03],
       [9.90041345e-01, 9.95865500e-03],
       [9.99993548e-01, 6.45246697e-06],
       [3.79838132e-03, 9.96201619e-01],
       [2.63217323e-03, 9.97367827e-01],
       [9.98028580e-01, 1.97141985e-03],
       [9.93169899e-01, 6.83010053e-03],
       [7.67137987e-03, 9.92328620e-01],
       [7.68852244e-04, 9.99231148e-01],
       [9.75689896e-01, 2.43101036e-02],
       [9.99969370e-01, 3.06303540e-05],
       [8.75618215e-01, 1.24381785e-01],
       [9.91613447e-01, 8.38655331e-03],
       [1.45234718e-01, 8.54765282e-01],
       [6.21149516e-02, 9.37885048e-01],
       [2.97075869e-03, 9.97029241e-01],
       [9.97211820e-01, 2.78818041e-03],
       [9.98639555e-01, 1.36044516e-03],
       [9.76200625e-01, 2.37993754e-02],
       [8.30316231e-01, 1.69683769e-01],
       [1.75291775e-01, 8.24708225e-01],
       [9.99525912e-01, 4.74088094e-04],
       [9.96267215e-01, 3.73278504e-03],
       [7.96842056e-03, 9.92031579e-01],
       [1.67147621e-01, 8.32852379e-01],
       [1.35011847e-01, 8.64988153e-01],
       [9.99999998e-01, 2.12569764e-09],
       [6.24294306e-01, 3.75705694e-01],
       [9.94365136e-01, 5.63486388e-03],
       [9.98094991e-01, 1.90500853e-03],
       [9.98822438e-01, 1.17756176e-03],
       [3.37138748e-01, 6.62861252e-01],
       [2.26380652e-03, 9.97736193e-01],
       [1.22974875e-01, 8.77025125e-01],
       [9.17510420e-01, 8.24895804e-02]])
#Оценка качества модели
balanced_accuracy_score(testY, LR.predict(testX))
0.9724506578947368
```

SVM

```
#Обучение модели
svm = SVC(kernel='rbf', probability=True)
svm.fit(trainX, trainY)
SVC(probability=True)
#Предсказание целевого признака
svm.predict_proba(testX)
array([[0.21664647, 0.78335353],
       [0.06797896, 0.93202104],
       [0.95024253, 0.04975747],
       [0.46058091, 0.53941909],
```

[0.22557141, 0.77442859],
[0.90904437, 0.09095563],
[0.21046708, 0.78953292],
[0.43194075, 0.56805925],
[0.8085475 , 0.1914525],
[0.48423372, 0.51576628],
[0.98509935, 0.01490065],
[0.3453356 , 0.6546644],
[0.15140535, 0.84859465],
[0.66908962, 0.33091038],
[0.97430626, 0.02569374],
[0.47461765, 0.52538235],
[0.30655768, 0.69344232],
[0.80565946, 0.19434054],
[0.04762935, 0.95237065],
[0.74295454, 0.25704546],
[0.86982156, 0.13017844],
[0.76978986, 0.23021014],
[0.71710924, 0.28289076],
[0.83093524, 0.16906476],
[0.7783003 , 0.2216997],
[0.18299455, 0.81700545],
[0.46218897, 0.53781103],
[0.98028763, 0.01971237],
[0.21681689, 0.78318311],
[0.88272612, 0.11727388],
[0.88921258, 0.11078742],
[0.09768064, 0.90231936],
[0.22405475, 0.77594525],
[0.8655315 , 0.1344685],
[0.11845655, 0.88154345],
[0.98290018, 0.01709982],
[0.23967676, 0.76032324],
[0.02372603, 0.97627397],
[0.11990345, 0.88009655],
[0.90225279, 0.09774721],
[0.04009535, 0.95990465],
[0.07879003, 0.92120997],
[0.35105664, 0.64894336],
[0.22773902, 0.77226098],
[0.60441577, 0.39558423],
[0.34877507, 0.65122493],
[0.63095669, 0.36904331],
[0.9675859 , 0.0324141],
[0.60441577, 0.39558423],
[0.39126792, 0.60873208],
[0.98006743, 0.01993257],
[0.95505093, 0.04494907],
[0.14112087, 0.85887913],
[0.15474579, 0.84525421],
[0.98355171, 0.01644829],

[0.95410898, 0.04589102],
[0.3514918 , 0.6485082],
[0.24192437, 0.75807563],
[0.42631079, 0.57368921],
[0.10365243, 0.89634757],
[0.80020718, 0.19979282],
[0.35197054, 0.64802946],
[0.27741081, 0.72258919],
[0.96427513, 0.03572487],
[0.22768279, 0.77231721],
[0.36839671, 0.63160329],
[0.72800551, 0.27199449],
[0.42745537, 0.57254463],
[0.89147666, 0.10852334],
[0.94346367, 0.05653633],
[0.04865892, 0.95134108],
[0.69130636, 0.30869364],
[0.91298286, 0.08701714],
[0.6285945 , 0.3714055],
[0.34611058, 0.65388942],
[0.04975238, 0.95024762],
[0.28266325, 0.71733675],
[0.98355136, 0.01644864],
[0.86407111, 0.13592889],
[0.55772528, 0.44227472],
[0.26357049, 0.73642951],
[0.83885472, 0.16114528],
[0.9677191 , 0.0322809],
[0.07153944, 0.92846056],
[0.7061842 , 0.2938158],
[0.89345255, 0.10654745],
[0.48565954, 0.51434046],
[0.91947508, 0.08052492],
[0.956124 , 0.043876],
[0.7438276 , 0.2561724],
[0.85644666, 0.14355334],
[0.90142624, 0.09857376],
[0.13142955, 0.86857045],
[0.25019906, 0.74980094],
[0.2239844 , 0.7760156],
[0.83667694, 0.16332306],
[0.188739 , 0.811261],
[0.59797919, 0.40202081],
[0.22893455, 0.77106545],
[0.98871146, 0.01128854],
[0.25920553, 0.74079447],
[0.2475924 , 0.7524076],
[0.91992634, 0.08007366],
[0.58658326, 0.41341674],
[0.13131151, 0.86868849],
[0.81446705, 0.18553295],

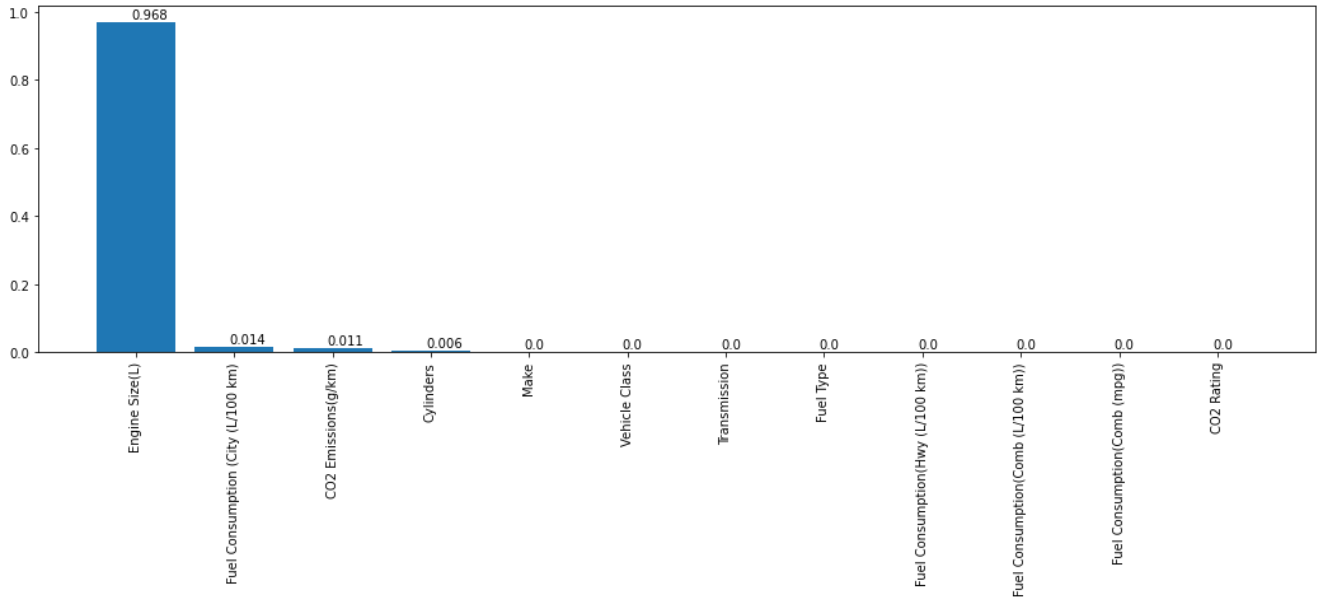
```
        [0.80927228, 0.19072772],
        [0.98087724, 0.01912276],
        [0.43854261, 0.56145739],
        [0.12055689, 0.87944311],
        [0.89482157, 0.10517843],
        [0.79853226, 0.20146774],
        [0.22821988, 0.77178012],
        [0.30682452, 0.69317548],
        [0.93272292, 0.06727708],
        [0.9950608 , 0.0049392 ],
        [0.77979716, 0.22020284],
        [0.79806636, 0.20193364],
        [0.60060816, 0.39939184],
        [0.21233766, 0.78766234],
        [0.0440081 , 0.9559919 ],
        [0.87254132, 0.12745868],
        [0.94934524, 0.05065476],
        [0.32866979, 0.67133021],
        [0.36612149, 0.63387851],
        [0.68036434, 0.31963566],
        [0.96836181, 0.03163819],
        [0.89550648, 0.10449352],
        [0.60847583, 0.39152417],
        [0.68910989, 0.31089011],
        [0.30793285, 0.69206715],
        [0.99609893, 0.00390107],
        [0.40290725, 0.59709275],
        [0.84436936, 0.15563064],
        [0.98831162, 0.01168838],
        [0.91828506, 0.08171494],
        [0.24174623, 0.75825377],
        [0.31564252, 0.68435748],
        [0.31581121, 0.68418879],
        [0.86035113, 0.13964887]])
#Оценка качества модели
balanced_accuracy_score(testY, svm.predict(testX))
0.8573190789473684
```

Деревья решений

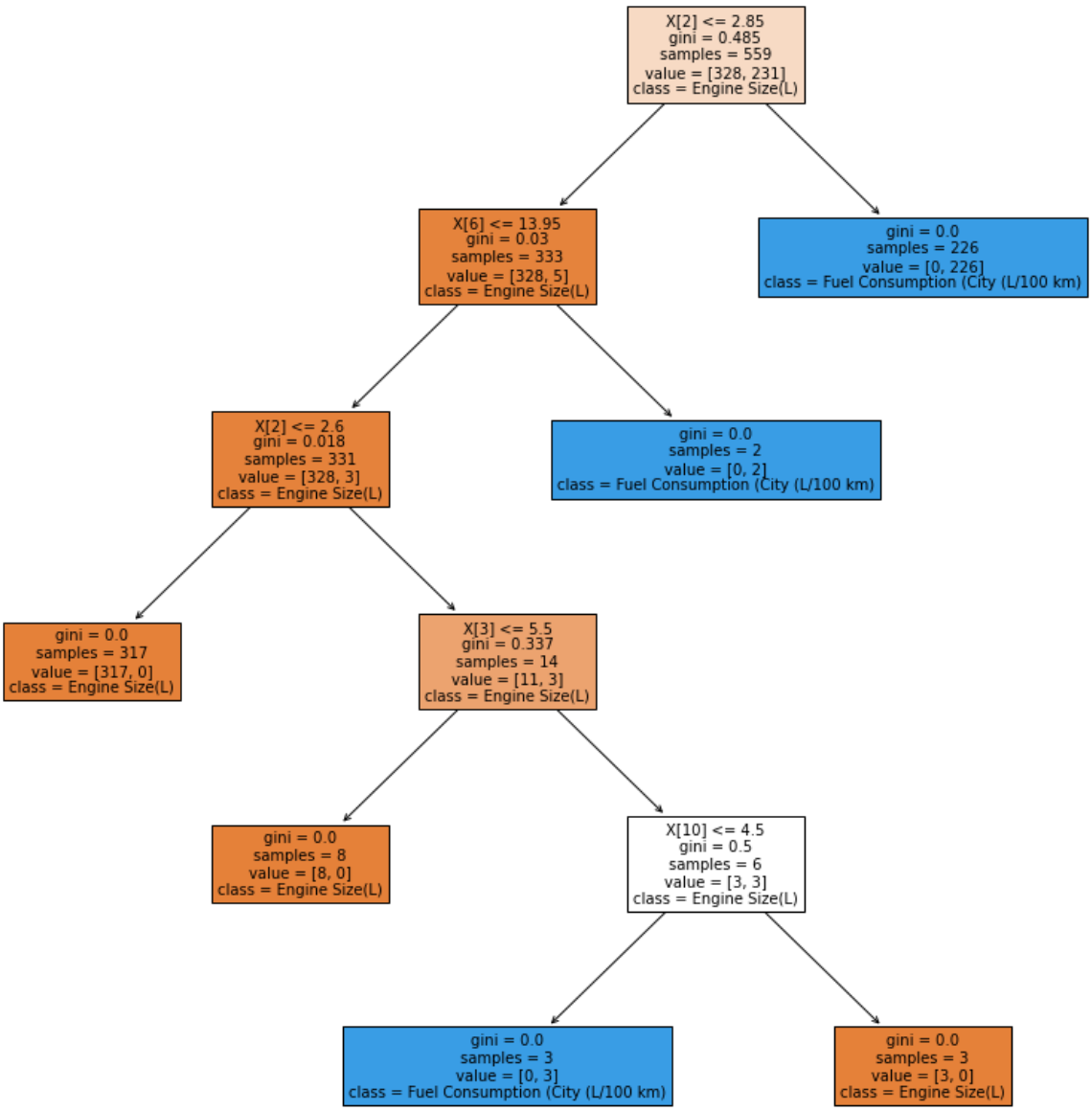
```
#Обучение модели
DeTree = DecisionTreeClassifier(random_state=1)
DeTree.fit(trainX, trainY)
DecisionTreeClassifier(random_state=1)
#Оценка качества модели
balanced_accuracy_score(testY, DeTree.predict(testX))
0.985608552631579
```

Как показала оценка качества модели, метод дерева решений работает наилучшим образом

```
#Функция построения графика для вывода признаков, наиболее
важных для определения целевого признака
def draw_feature_importances(tree_model, X_dataset,
figsize=(18,5)):
    """
    Вывод важности признаков в виде графика
    """
    # 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
diagram, _ = draw_feature_importances(DeTree, data)
```




```
#Визуальное отображение дерева решений
fig, ax = plt.subplots(figsize=(15, 15))
cn=['Engine Size(L)', 'Fuel Consumption (City (L/100 km)', 'CO2
Emissions(g/km', 'Cylinders' ]
tree.plot_tree(DeTree, fontsize=10,class_names=cn, filled=True)
plt.show()
```


X[2] <= 2.85
gini = 0.485
samples = 559
value = [328, 231]
class = Engine Size(L)
X[6] <= 13.95
gini = 0.03
samples = 333
value = [328, 5]
class = Engine Size(L)
gini = 0.0
samples = 226
value = [0, 226]
class = Fuel Consumption (City (L/100 km)
X[2] <= 2.6
gini = 0.018
samples = 331
value = [328, 3]
class = Engine Size(L)
gini = 0.0
samples = 2
value = [0, 2]
class = Fuel Consumption (City (L/100 km)
gini = 0.0
samples = 317
value = [317, 0]
class = Engine Size(L)
X[3] <= 5.5
gini = 0.337
samples = 14
value = [11, 3]
class = Engine Size(L)
gini = 0.0
samples = 8
value = [8, 0]
class = Engine Size(L)
X[10] <= 4.5
gini = 0.5
samples = 6
value = [3, 3]
class = Engine Size(L)
gini = 0.0
samples = 3
value = [0, 3]
class = Fuel Consumption (City (L/100 km)
gini = 0.0
samples = 3
value = [3, 0]
class = Engine Size(L)