Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждениевысшего образования
«Московский государственный технический университетимени Н.Э. Баумана (национальный исследовательский университет)»(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

Отчет по рубежному контролю №2

Вариант 16

По дисциплине:
«Технологии машинного обучения»

Выполнил:

Студент группы ИУ5 ________________ Наказной Н.А.

(Подпись, дата) (Фамилия И.О.)

Проверил:

_____________________ **Гапанюк Ю. Е.**

(Подпись, дата) (Фамилия И.О.)

Москва, 2022

Задание. Для заданного набора данных (по варианту) построить модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей использовать дерево решений и случайный лес. Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

- При решении задач можно выбирать любое подмножество признаков из приведенного набора данных.
- Для сокращения времени построения моделей можно использовать фрагмент набора данных (например, первые 200-500 строк).

Выполнение задания

Импорт библиотек

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Загрузка и первичная обработка данных:

```
# загрузка набора данных
data = pd.read_csv('restaurant-scores-lives-standard.csv', sep=",")
# размер набора данных
data.shape
```

```
(53973, 23)
```

```
# первые 5 строк набора данных
data.head()
```

	business_id	business_name	business_address	business_city	business_state	business_postal_code	business_latitude	business_longitude	business_location	business_phone_nu
0	101192	Cochinita #2	2 Marina Blvd Fort Mason	San Francisco	CA	NaN	NaN	NaN	NaN	1.415043
1	97975	BREADBELLY	1408 Clement St	San Francisco	CA	94118	NaN	NaN	NaN	1.415724
2	92982	Great Gold Restaurant	3161 24th St.	San Francisco	CA	94110	NaN	NaN	NaN	
3	101389	HOMAGE	214 CALIFORNIA ST	San Francisco	CA	94111	NaN	NaN	NaN	1.415488
4	85986	Pronto Pizza	798 Eddy St	San Francisco	CA	94109	NaN	NaN	NaN	

5 rows × 23 columns

```
# проверим, есть ли пропущенные значения
data.isnull().sum()
```

```
business_id                    0
business_name                  0
business_address               0
business_city                  0
business_state                 0
business_postal_code        1018
business_latitude          19556
business_longitude         19556
business_location          19556
business_phone_number      36938
inspection_id                  0
inspection_date                0
inspection_score           13610
inspection_type                0
violation_id               12870
violation_description      12870
risk_category              12870
Neighborhoods (old)        19594
Police Districts           19594
Supervisor Districts       19594
Fire Prevention Districts  19646
Zip Codes                  19576
Analysis Neighborhoods     19594
dtype: int64
```

```
# уникальные значения столбца 'category_group' файла impeachment_topline
data['Analysis Neighborhoods'].unique()
```

```
array([nan, 34., 36.,  9., 23., 20.,  8., 25.,  1., 13., 35., 32., 39.,
       12., 26., 22.,  7.,  6., 10., 14.,  5., 21., 29., 28., 11., 30.,
        2.,  3., 15.,  4., 31., 18., 41., 16., 24., 27., 40., 17., 19.,
       33., 37., 38.])
```

```
# удаление колонок неподходящих для построения моделей
data.drop(['business_id', 'business_name', 'business_address', 'business_city', 'business_state', 'business_postal_code', 'business_latitude', 'busi
```

```
data.head()
```

	inspection_score	violation_description	risk_category	Neighborhoods (old)	Police Districts	Supervisor Districts	Fire Prevention Districts	Analysis Neighborhoods
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	96.0	Inadequately cleaned or sanitized food contact...	Moderate Risk	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	High risk vermin infestation	High Risk	NaN	NaN	NaN	NaN	NaN

```
# удаление строк, содержащих пустые значения в колонке целевого признака
data.dropna(axis=0, subset=['Analysis Neighborhoods'], inplace=True)
# размер данных
data.shape
```

```
(34379, 8)
```

```
data.head()
```

	inspection_score	violation_description	risk_category	Neighborhoods (old)	Police Districts	Supervisor Districts	Fire Prevention Districts	Analysis Neighborhoods
11	71.0	Improper storage use or identification of toxi...	Low Risk	34.0	2.0	9.0	6.0	34.0
16	84.0	Moderate risk food holding temperature	Moderate Risk	36.0	9.0	9.0	7.0	36.0
30	NaN	NaN	NaN	10.0	9.0	11.0	7.0	9.0
55	NaN	Unapproved or unmaintained equipment or utensils	Low Risk	36.0	9.0	9.0	7.0	36.0
64	92.0	Inadequate and inaccessible handwashing facili...	Moderate Risk	23.0	1.0	10.0	3.0	23.0

```
# проверим, есть ли пропущенные значения
data.isnull().sum()
```

```
inspection_score             7262
violation_description        7232
risk_category                7232
Neighborhoods (old)             0
Police Districts                0
Supervisor Districts            0
Fire Prevention Districts      52
Analysis Neighborhoods          0
dtype: int64
```

```
# удаление строк, содержащих пустые значения в колонках
data.dropna(axis=0, subset=['Police Districts'], inplace=True)
data.dropna(axis=0, subset=['violation_description'], inplace=True)
data.dropna(axis=0, subset=['inspection_score'], inplace=True)
data.dropna(axis=0, subset=['Fire Prevention Districts'], inplace=True)
# размер данных
data.shape
```

```
(25812, 8)
```

[ ]:
```
data.head()
```

[ ]:

	inspection_score	violation_description	risk_category	Neighborhoods (old)	Police Districts	Supervisor Districts	Fire Prevention Districts	Analysis Neighborhoods
11	71.0	Improper storage use or identification of toxi...	Low Risk	34.0	2.0	9.0	6.0	34.0
16	84.0	Moderate risk food holding temperature	Moderate Risk	36.0	9.0	9.0	7.0	36.0
64	92.0	Inadequate and inaccessible handwashing facili...	Moderate Risk	23.0	1.0	10.0	3.0	23.0
73	92.0	Moderate risk food holding temperature	Moderate Risk	34.0	2.0	9.0	12.0	34.0
92	74.0	Foods not protected from contamination	Moderate Risk	6.0	1.0	10.0	3.0	8.0

[ ]:
```
# проверим, есть ли пропущенные значения
data.isnull().sum()
```

[ ]:
```
inspection_score             0
violation_description        0
risk_category                0
Neighborhoods (old)          0
Police Districts             0
Supervisor Districts         0
Fire Prevention Districts    0
Analysis Neighborhoods       0
dtype: int64
```

:
```
#Consolidate Types of Violation
hygiene_v = dict.fromkeys(['Unclean or degraded floors walls or ceilings', 'Wiping cloths not clean or properly stored or inadequate sanitizer', 'Mo
infralack_v = dict.fromkeys(['Inadequate and inaccessible handwashing facilities', 'Inadequate or unsanitary refuse containers or area or no garbage
legal_v = dict.fromkeys(['Food safety certificate or food handler card not available', 'Unapproved or unmaintained equipment or utensils', 'Permit l
noncompliance_v = dict.fromkeys(['High risk food holding temperature', 'Inadequate food safety knowledge or lack of certified food safety manager',
data = data.replace(hygiene_v)
data = data.replace(infralack_v)
data = data.replace(legal_v)
data = data.replace(noncompliance_v)
```


:
```
#Consolidate Types of Violation
hygiene_v = dict.fromkeys(['Unclean or degraded floors walls or ceilings', 'Wiping cloths not clean or properly stored or inadequate sanitizer', 'Mo
infralack_v = dict.fromkeys(['Inadequate and inaccessible handwashing facilities', 'Inadequate or unsanitary refuse containers or area or no garbage
legal_v = dict.fromkeys(['Food safety certificate or food handler card not available', 'Unapproved or unmaintained equipment or utensils', 'Permit l
noncompliance_v = dict.fromkeys(['High risk food holding temperature', 'Inadequate food safety knowledge or lack of certified food safety manager',
data = data.replace(hygiene_v)
data = data.replace(infralack_v)
data = data.replace(legal_v)
data = data.replace(noncompliance_v)
```


:
```
data.head()
```

[ ]:

	inspection_score	violation_description	risk_category	Neighborhoods (old)	Police Districts	Supervisor Districts	Fire Prevention Districts	Analysis Neighborhoods
11	71.0	Noncompliance	Low Risk	34.0	2.0	9.0	6.0	34.0
16	84.0	Noncompliance	Moderate Risk	36.0	9.0	9.0	7.0	36.0
64	92.0	Lack Infrastructure	Moderate Risk	23.0	1.0	10.0	3.0	23.0
73	92.0	Noncompliance	Moderate Risk	34.0	2.0	9.0	12.0	34.0
92	74.0	Hygiene	Moderate Risk	6.0	1.0	10.0	3.0	8.0

[ ]:
```
# Выбираем случайные 500 строк для сокращения времени построения моделей
data.sample(n = 500)
```

[ ]:

	inspection_score	violation_description	risk_category	Neighborhoods (old)	Police Districts	Supervisor Districts	Fire Prevention Districts	Analysis Neighborhoods
25281	68.0	Noncompliance	High Risk	41.0	9.0	1.0	13.0	39.0
16111	91.0	Noncompliance	High Risk	6.0	2.0	9.0	6.0	8.0
29317	94.0	Noncompliance	Low Risk	17.0	9.0	1.0	13.0	13.0
38903	98.0	Noncompliance	Low Risk	2.0	7.0	7.0	2.0	2.0
25926	92.0	Noncompliance	Moderate Risk	3.0	4.0	5.0	15.0	5.0
...	...	...	...	...	...	...	...	...
47161	90.0	Lack Infrastructure	Moderate Risk	2.0	7.0	7.0	2.0	2.0
8356	88.0	Hygiene	Low Risk	1.0	3.0	8.0	10.0	1.0
38707	96.0	Hygiene	Moderate Risk	6.0	1.0	10.0	4.0	8.0

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
category = le.fit_transform(data['risk_category'])
discr = le.fit_transform(data['violation_description'])
data['risk_category'].unique()
```

```
array(['Low Risk', 'Moderate Risk', 'High Risk'], dtype=object)
```

```
np.unique(category)
```

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

```
data['violation_description'].unique()
```

```
array(['Noncompliance', 'Lack Infrastructure', 'Hygiene', 'Legal'],
      dtype=object)
```

```
np.unique(discr)
```

```
array([0, 1, 2, 3])
```

```
data['risk_category']=category
data['violation_description']=discr
data.head()
```

	inspection_score	violation_description	risk_category	Neighborhoods (old)	Police Districts	Supervisor Districts	Fire Prevention Districts	Analysis Neighborhoods
11	71.0	3	1	34.0	2.0	9.0	6.0	34.0
16	84.0	3	2	36.0	9.0	9.0	7.0	36.0
64	92.0	1	2	23.0	1.0	10.0	3.0	23.0
73	92.0	3	2	34.0	2.0	9.0	12.0	34.0
92	74.0	0	2	6.0	1.0	10.0	3.0	8.0

```
# Масштабирование данных
from sklearn.preprocessing import MinMaxScaler
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['inspection_score']])
data['inspection_score'] = sc1_data
sc2_data = sc1.fit_transform(data[['Neighborhoods (old)']])
data['Neighborhoods (old)'] = sc2_data
sc3_data = sc1.fit_transform(data[['Police Districts']])
data['Police Districts'] = sc3_data
sc4_data = sc1.fit_transform(data[['Supervisor Districts']])
data['Supervisor Districts'] = sc4_data
sc5_data = sc1.fit_transform(data[['Fire Prevention Districts']])
data['Fire Prevention Districts'] = sc5_data
sc6_data = sc1.fit_transform(data[['violation_description']])
data['violation_description'] = sc6_data
sc7_data = sc1.fit_transform(data[['risk_category']])
data['risk_category'] = sc7_data
sc8_data = sc1.fit_transform(data[['Analysis Neighborhoods']])
data['Analysis Neighborhoods'] = sc8_data
data.head()
```

	inspection_score	violation_description	risk_category	Neighborhoods (old)	Police Districts	Supervisor Districts	Fire Prevention Districts	Analysis Neighborhoods
11	0.462963	1.000000	0.5	0.825	0.111111	0.8	0.357143	0.825
16	0.703704	1.000000	1.0	0.875	0.888889	0.8	0.428571	0.875
64	0.851852	0.333333	1.0	0.550	0.000000	0.9	0.142857	0.550
73	0.851852	1.000000	1.0	0.825	0.111111	0.8	0.785714	0.825
92	0.518519	0.000000	1.0	0.125	0.000000	0.9	0.142857	0.175

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

```
# Разделение данных на тестовую и обучающую выборки
data_train, data_test, data_y_train, data_y_test = train_test_split(data[data.columns.drop('Analysis Neighborhoods')], data['Analysis Neighborhoods'
```

Модель "Дерево решений"

```
from sklearn.tree import DecisionTreeRegressor
dtc = DecisionTreeRegressor(random_state=1).fit(data_train, data_y_train)
data_test_predicted_dtc = dtc.predict(data_test)
```

Модель "Случайный лес"

```
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor(random_state=1).fit(data_train, data_y_train)
data_test_predicted_rf = RF.predict(data_test)
```

Оценка качества моделей:

В качестве метрик для оценки качества моделей используем Mean squared error (средняя квадратичная ошибка), как наиболее часто используемую метрику для оценки качества регрессии, и метрику R^2 (коэффициент детерминации), потому что эта метрика является нормированной.

```
from sklearn.metrics import mean_squared_error, r2_score
# Mean squared error - средняя квадратичная ошибка
print('Метрика MSE:\nДерево решений: {}\nСлучайный лес: {}'.format(mean_squared_error(data_y_test, data_test_predicted_dtc), mean_squared_error(data_
```

```
Метрика MSE:
Дерево решений: 7.748334108166603e-07
Случайный лес: 3.951650395165032e-07
```

```
# 4) Метрика R2 или коэффициент детерминации
print('Метрика R\u00B2:\nДерево решений: {}\nСлучайный лес: {}'.format(r2_score(data_y_test, data_test_predicted_dtc), r2_score(data_y_test, data_tes
```

```
Метрика R²:
Дерево решений: 0.9999907284691102
Случайный лес: 0.9999952715192462
```

Выводы о качестве построенных моделей:

Исходя из результатов первой метрики, можно сделать вывод что модель "Случайный лес" лучше справляется с задачей по сравнению с моделью "Дерево решений". По результатам второй метрики можно сказать, что переменные практически функционально зависимы.