МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по индивидуальному домашнему заданию

по дисциплине «Машинное обучение»

Тема: Построение регрессионной модели для предсказания рейтинга

Студент гр. 6307 _____ Трофимов Н.И.

Преподаватель _____ Жангиров Т.Р

Санкт-Петербург

2020

Цель работы.

Датасет содержит информацию о шоколаде. Датасет содержит 1795 наблюдений. Один из признаков отвечает за рейтинг шоколада.

Задача заключается в выделение значимых признаков и построении регрессионной и классификационной моделей для предсказания рейтинга.

Ход работы

Каждый шоколад оценивается на основе сочетания объективных качеств и субъективной интерпретации. База данных шоколада сосредоточена на темном шоколаде с целью оценить вкус какао, превращенного в шоколад.

Был выгружен csv-файл в DataFrame, результат представлен на рисунке 1.

Company \n(Maker-if known)	Specific Bean Origin\nor Bar Name	REF	Review\nDate	Cocoa\nPercent	Company\nLocation	Rating	Bean\nType	Broad Bean\nOrigin
A. Morin	Agua Grande	1876	2016	63%	France	3.75		Sao Tome
A. Morin	Kpime	1676	2015	70%	France	2.75		Togo
A. Morin	Atsane	1676	2015	70%	France	3.00		Togo
A. Morin	Akata	1680	2015	70%	France	3.50		Togo
A. Morin	Quilla	1704	2015	70%	France	3.50		Peru

Рисунок 1. Датафрейм данных

Описание данных:

- Company (Maker if known) – название компании-производителя
- Specific Bean Origin or Bar name – место происхождения бабов
- REF – уникальный идентификатор
- Review Date – год публикации отзыва
- Cocoa Percent – процент какао в шоколаде
- CompanyLocation – страна производитель
- Rating – рейтинг шоколада
- Bean Type – сорт бобов

- Broad BeanOrigin – место происхождения бобов

Для дальнейшего удобства работы с данными были переименованы колонки, а также добавлена новая колонка типа float64 для представления процентного содержания какао.

На рисунке 2 представлен график рейтинга шоколада.

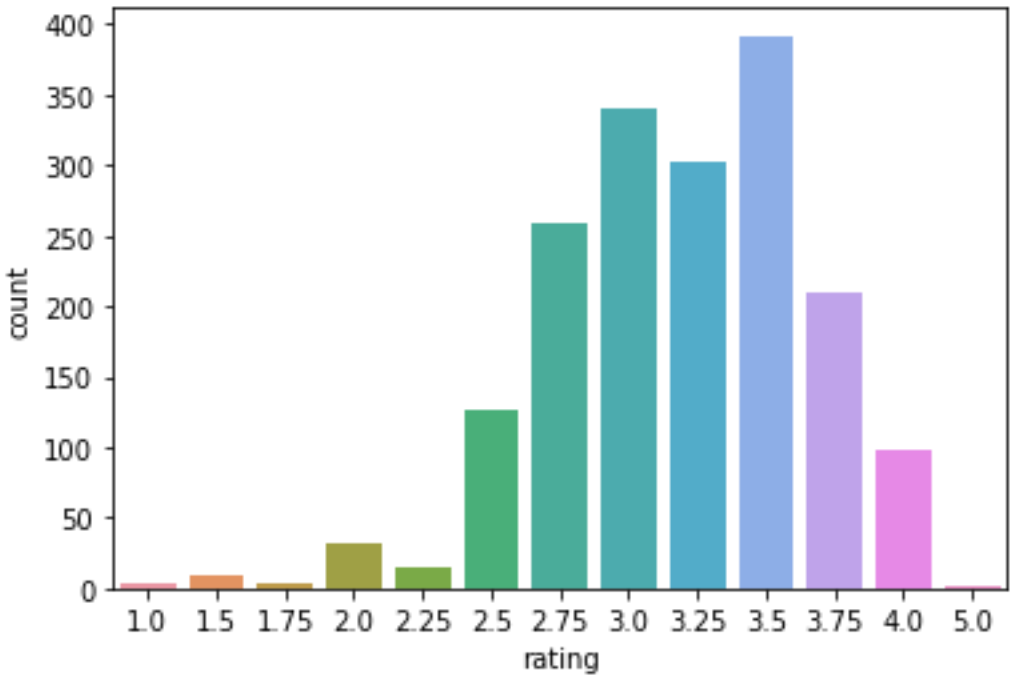

Рисунок 2. Рейтинг шоколада

На рисунке 3 представлен график зависимости рейтинга шоколада от процента содержащегося в нем какао бобов.

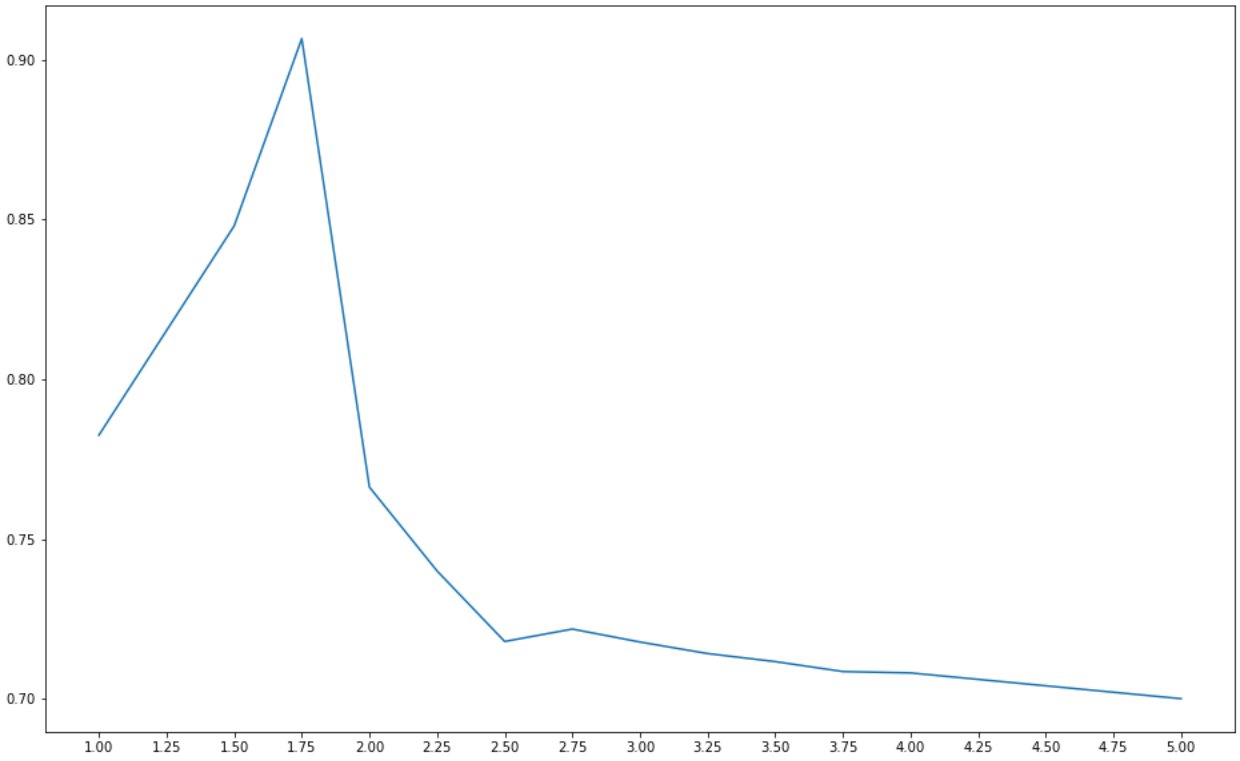

Рисунок 3. Зависимость рейтинга шоколада от процента

Исходный датасет в основном состоит из колонок типа object, которые содержат категориальные переменные (текстовые данные), эти данные представлены на рисунке 4.

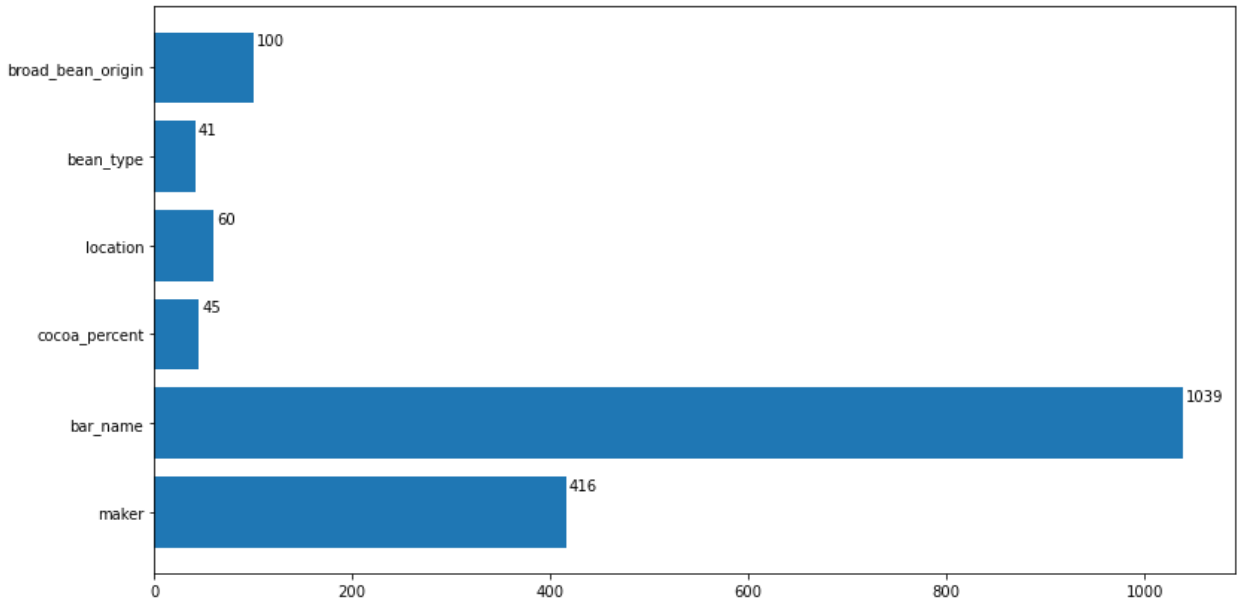

Рисунок 4. Уникальные значения дли колонок типа object

Для дальнейшей работы с данными необходимо закодировать данные в колонках строковых типов данных. Для колонки “bar_name” был применен метод LabelEncoder, который даст каждому уникальному строковому значению колонки уникальное целочисленное значение (некоторый id).

Для остальных колонок, кроме “cocoa_percent” (эта колонка была отброшена) был применен метод OneHotEncoder. Данный метод создает новую матрицу, где каждая колонка это закодированное уникальное строковое значение, а значение в этой колонке 1 или 0 показывает – присуще ли это значение данной строке или нет (имеется ли это свойство).

Затем были отброшены 2 колонки “ref”, “review_date” поскольку они сильно коррелируют между собой, а также не несут полезной информации для обучения. Результирующий датасет имеет размер *1795x620* и представлен на рисунке 5.

bar_name	rating	percent	0	1	2	3	4	5	6	...	607	608	609	610	611	61
14	3.75	0.63	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
492	2.75	0.70	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
67	3.00	0.70	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
15	3.50	0.70	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
809	3.50	0.70	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
754	3.75	0.70	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
258	3.00	0.65	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
483	3.50	0.65	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
483	3.25	0.62	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
137	3.00	0.65	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

Рисунок 5. Датасет для обучения

Из данного датасета была выделена колонка “rating”, которая стала целью, которую надо предсказать.

Были обучены две модели – классификации и регрессии. Для модели классификации рейтинг был преобразован в целочисленные значения, т.о. он мог быть только 1, 2, 3, 4 или 5.

Обученная на этих данных модель классификации показала результат с точностью 72.4%. Предсказанные ею классы изображены на рисунке 6.

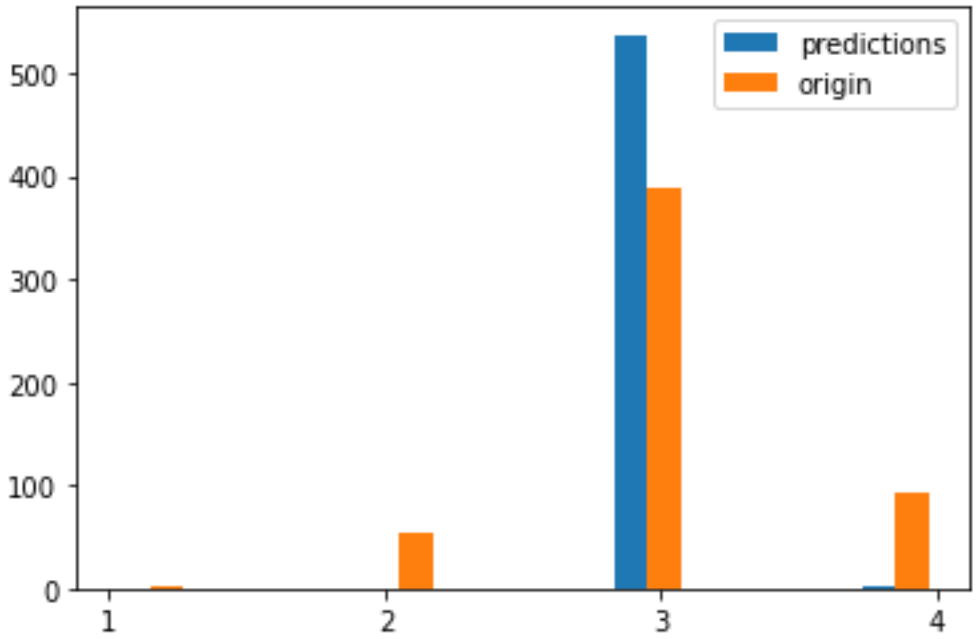

Рисунок 6. Результат модели классификации

Были подсчитаны метрики “accuracy”, “precision”, “recall”, “f1”. Результаты представлены в таблице 1.

Table 1. Метрики модели классификации

accuracy	precision	recall	F1
0.724	0.695	0.724	0.61

На рисунке 7 изображены показатели TruePositive, FalseNegative, FalsePositive для каждого класса.

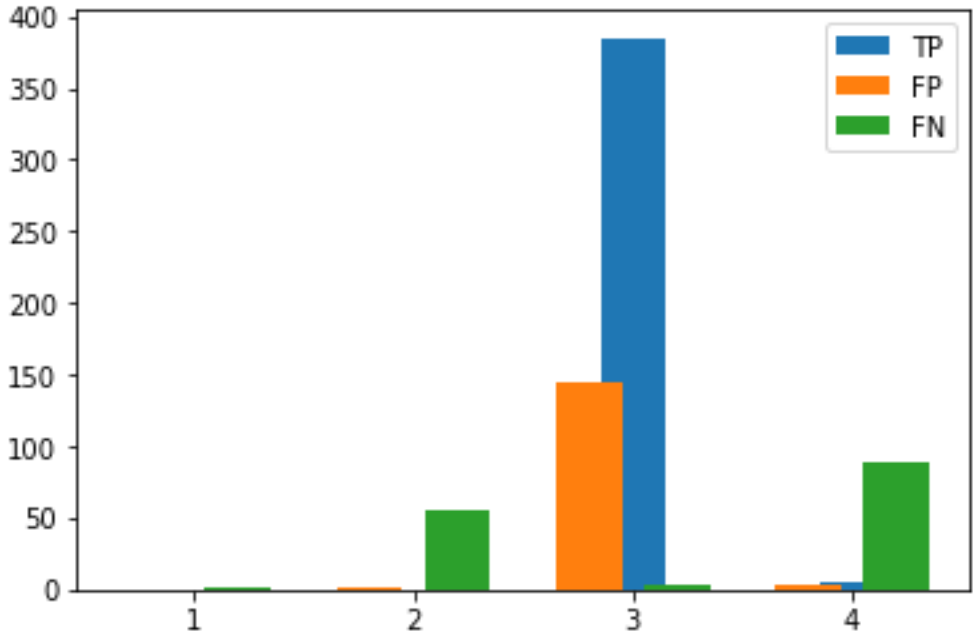

Рисунок 7. TP, FP, FN для каждого класса

Так же на этих данных была обучена регрессионная модель, результаты который были округлены по заданным правилам. Результаты данной модели представлены в таблице 2.

MAE	MSE	MAX_ERROR
0.45	0.45	3.5

ВЫВОД

В данной работе были построены модели классификации и регрессии для предсказания класса рейтинга шоколада и самого рейтинга.

Исходный набор данных представляет краткую информацию о шоколаде, а также данные, относящиеся к отзыву на шоколад. Большая часть данных имеет строковый тип данных, что в свою очередь потребовало предобработки данных – кодирование значений. В исходном датасете были данные, которые не несут полезной информации – дата создания отзыва и его уникальный номер. Возможно, с их наличием модели предсказали бы результат лучше, однако они все же были удалены из тренировочного и тестового наборов.

Модель классификации дала результат выше, чем модель регрессии. Это связано в первую очередь с нехваткой данных (фичей) и общего объема данных, некоторых классов было довольно мало.

Возникшие проблемы

1. Низкий результат моделей регрессии и классификации. *Решение*: можно заняться дополнительной настройкой данных. Некоторые данные состоят из списка значений, которые можно было бы также разделить на различные фичи. В добавок можно добавить больше данных, которые несут смысловую нагрузку, например – состав шоколада.
2. Неоднозначность предсказания модели регрессии. Результаты отклонялись от ожидаемых на несколько десятых(сотых). Как решение была написана функция, округляющая поученные предсказания модели. Возможно ее стоит настроить лучше.

ИСХОДНЫЙ КОД

```
from collections import Counter
from math import modf

import pandas as pd
import numpy as np


import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, confusion_matrix
from sklearn.metrics import max_error, mean_absolute_error, mean_squared_error

data = pd.read_csv("idz_data/flavors_of_cacao.csv")

new_columns = [
    "maker",
    "bar_name",
    "ref",
    "review_date",
    "cocoa_percent",
```

```
        "location",
        "rating",
        "bean_type",
        "broad_bean_origin"
    ]

    data.columns = new_columns
    data.head()

    data["percent"] = data.cocoa_percent.apply(lambda x: float(x.split("%")[0])/100)

    data.bean_type = data.bean_type.fillna('\xa0')
    data.broad_bean_origin = data.broad_bean_origin.fillna('\xa0')

    s = (data.dtypes == 'object')
    object_columns = list(s[s].index)
    object_columns.remove("bar_name")
    object_columns.remove("cocoa_percent")
    # object_columns.pop(object_columns.index("bean_type"))

    # ONE HOT ENCODING and LABEL ENCODER


    encoded_data = data.copy()

    label_encoder = LabelEncoder()
```

```
    OH_encoder = OneHotEncoder(sparse=False)

    encoded_data["bar_name"] =
label_encoder.fit_transform(encoded_data["bar_name"])


    data_encoded_matrix = pd.DataFrame(
        OH_encoder.fit_transform(encoded_data[object_columns]),
    )

    data_numeric = encoded_data.drop(object_columns, axis=1)

    encoded_data = pd.concat([data_numeric, data_encoded_matrix], axis=1)

    encoded_data = encoded_data.drop(columns=["ref", "review_date",
"cocoa_percent"])
    X, Y = encoded_data.drop(columns=["rating"]), encoded_data["rating"]
    def round_rating_75(rating):
        if rating < 1.75:
            return 1
        elif rating < 2.75:
            return 2
        elif rating < 3.75:
            return 3
        elif rating < 4.75:
            return 4
        return 5
```

```
def round_rating(rating):
    if rating < 2:
        return 1
    if rating < 3:
        return 2
    if rating < 4:
        return 3
    if rating < 5:
        return 4
    return 5

Y_rounded = Y.apply(round_rating_75)

def round_rating_regression(rating):
    remainder, integer = modf(rating)

    if integer < 5:
        if remainder < 0.125:
            return int(integer)
        if remainder <= 0.25:
            value = f"{int(integer)}.25"
            return float(value)
        if remainder <= 0.5:
            value = f"{int(integer)}.5"
            return float(value)
```

```
        if remainder <= 0.75:
            value = f"{int(integer)}.75"
            return float(value)
    return 5
    x_train, x_test, y_train, y_test = train_test_split(
    X,
    Y_rounded,
    test_size=0.3,
    random_state=1
)
forest_ = RandomForestClassifier()

parameters = {
    "n_estimators": [*range(2, 150, 12)],
    "max_depth": [*range(3, 12)],
    "min_samples_leaf": [1, 2, 3, 4, 5],
#     "n_jobs": [-1],



}


grid = GridSearchCV(
    forest_,
    parameters,
    cv=5
)
```

```
grid.fit(x_train, y_train)
random_forest_classifier = RandomForestClassifier(
    n_estimators=2,
    min_samples_leaf=3,
    max_depth=11,
    n_jobs=-1,
    random_state=12
)

random_forest_classifier.fit(x_train, y_train)

random_forest_classifier.score(x_test, y_test)
predictions = random_forest_classifier.predict(x_test)

accuracy = accuracy_score(y_test, predictions)
precision, recall, f1, _ = precision_recall_fscore_support(
    y_test,
    predictions,
    average="weighted",
    zero_division=0
)

metrics_df = pd.DataFrame(
    {
        "accuracy": [accuracy],
        "precision": [precision],
```

```
        "recall": [recall],
        "f_measure": [f1]
    }
)

random_forest_regressor = RandomForestRegressor(
    n_estimators=62,
    min_samples_leaf=2,
    max_depth=11,
    n_jobs=-1,
    random_state=12
)

random_forest_regressor.fit(x_train, y_train)

random_forest_regressor.score(x_test, y_test)

predictions = random_forest_regressor.predict(x_test)

predictions = list(map(round_rating_regression, predictions))

metrics_df = pd.DataFrame(
    {
        "mae": [mean_absolute_error(y_test, predictions)],
        "mse": [mean_squared_error(y_test, predictions)],
        "max_error": [max_error(y_test, predictions)],
    })
```