Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования «Рыбинский государственный

авиационный технический университет имени П. А. Соловьева»

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ УПРАВЛЕНИЯ

Математического и программного обеспечения электронных

вычислительных средств

ОТЧЁТ

по дисциплине:

«Методы и алгоритмы анализа данных»

на тему:

«Основы работы с библиотекой *scikit-learn*»

Выполнил: студент группы ИВМ-24 Морозов А. А.

Руководитель: ассистент Вязниковцев Д. А.

Рыбинск 2024

Содержание

[Цель работы 3](#_Toc179403870)

[1. Работа с библиотекой *scikit-learn* 4](#_Toc179403871)

[Вывод 9](#_Toc179403872)

# Цель работы

Целями данной лабораторной работы являются:

- изучение основ работы с библиотекой *scikit-learn*;

- подготовить данные и провести машинное обучение разными алгоритмами *scikit-learn*;

- построить графики с помощью библиотеки *matplotlib*.

# 1. Работа с библиотекой *scikit-learn*

Перед началом работы с библиотекой *scikit-learn* необходимо отредактировать начальный *dataframe*:

df = pd.read\_csv('train.csv')  
df = df.drop(columns=['dropoff\_datetime'])  
df = df.sort\_values(by="pickup\_datetime")  
train\_df, test\_df = df.iloc[:1000000], df.iloc[1000000:]

После выполнения кода были созданы 2 *dataframe* (один для обучения модели, а второй для проверки работоспособности модели).

Далее выведем на экран ненормализованную гистограмму (рисунок 1) и нормализованную (рисунок 2). Нормализация происходила с помощью функции *numpy.log1p()*

Изображение выглядит как текст, Прямоугольник, снимок экрана, линия

Автоматически созданное описание

Рисунок 1 – Ненормализованная гистограмма

Изображение выглядит как диаграмма, снимок экрана, График, линия

Автоматически созданное описание

Рисунок 2 – Нормализованная гистограмма

Добавим результат нормализации как новый столбец с названием *log\_trip\_duration* в тестовую и обучающую выборки и преобразуем его в нужны формат времени с помощью функции *pandas.to\_datetime*.

Визуализируем число поездок по дням (рисунок 3).

Изображение выглядит как снимок экрана, линия, График

Автоматически созданное описание

Рисунок 3 – График числа поездок по дням

Выполним группировку обучающей выборки по значению *dates* используя функцию *groupby*(), которую применим к обучающему *dataframe.*

Соединим столбики дня недели и часа поездки:

def create\_features(data\_frame):

X = pd.concat(

[

data\_frame.pickup\_datetime.apply(lambda x: x.timetuple().tm\_yday),

data\_frame.pickup\_datetime.apply(lambda x: x.hour)

], axis=1, keys=['day', 'hour']

)

return X, data\_frame.log\_trip\_duration

Применим вышеописанную функцию *create\_features*() для формирования *X\_train*, *y\_train* и *X\_test*, *y\_test* и выведем последние 5 записей в *dataframe* *X\_train* (рисунок 4)

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 4 – Последние 5 записей в *X\_train*

Преобразуем признак *hour* в категориальный (функция *ColumnTransformer*()) и применим его для изменения обучающей выборки (функция *fit\_transform*).

После изменения в dataframe X\_train теперь 25 столбиков (на каждый час в сутках).

Теперь можно обучить модель с помощью алгоритма линейной регрессии, а также выполнить предсказание и найти среднеквадратическую ошибку алгоритма:

def LinePredict(X\_train, y\_train, X\_test, y\_test):

line = LinearRegression()

line.fit(X=X\_train, y=y\_train)

predict = line.predict(X\_test)

print(f"Предсказание у линейной регрессии: {predict}")

print(f"MSE: {mean\_squared\_error(y\_test, predict)}")

На рисунке 5 изображён вывод в консоли после выполнения функции *LinePredict*()



Рисунок 5 – Метод линейной регрессий

Для обучения с помощью алгоритма ближайших соседей воспользуемся кодом:

def Neighbors(X\_train, y\_train, X\_test, y\_test):

knn = KNeighborsRegressor(n\_neighbors=5, weights='uniform', p=2)

knn.fit(X\_train, y\_train)

predictions = knn.predict(X\_test)

print(f"Предсказание у методов основанных на ближайших соседях: {predictions}")

print(f"MSE: {mean\_squared\_error(y\_test, predictions)}")

На рисунке 6 изображён вывод в консоли после выполнения функции *Neighbors*()



Рисунок 6 – Метод ближайших соседей

Из рисунков 5 и 6 следует то, что у метода ближайших соседей среднеквадратическая ошибка больше, чем у метода линейной регрессии почти в 2 раза. Также для обучения используя метод ближайших соседей необходимо выделять больше системных ресурсов и времени чем у метода линейно регрессии.

# Вывод

В результате выполнения лабораторной работы были изучены основы работы с библиотекой *scikit-learn*. Произведено машинное обучение с помощью методов линейной регрессии и ближайших соседей. В результате выяснено преимущество метода линейной регрессии над методом ближайших соседей. Построены графики с помощью библиотеки *matplotlib*.