**Проект по предмету Машинное обучение по датасету: «New York City Taxi Fare Prediction»**

В рамках данного проекта использовался датасет доступный по ссылке с соревнования на Kaggle, доступный по ссылке: <https://www.kaggle.com/competitions/new-york-city-taxi-fare-prediction/data>.

Для формирования модели машинного обучения использовался файл train.csv. Файл состоит из 8 колонок, краткое описание представлено ниже:

* Key – уникальная строка, идентифицирующая каждую строку. Состоит из pickup\_datetime плюс уникальной целое число;
* Pickup\_datetime – временная метка, указывающая время начала поездки на такси;
* Pickup\_longitude – значение координаты долготы начала поездки на такси;
* Pickup\_latitude – значение координаты долготы окончания поездки на такси;
* Dropoff\_longitude – значение координаты долготы окончания поездки на такси;
* Dropoff\_latitude – значение координаты широты окончания поездки на такси;
* Passenger\_count – целое число, указывающее количество пассажиров в поездке на такси;
* Fare\_amount – долларовая сумма стоимости поездки на такси.

В общем датасете используется порядка 55 миллионов строчек, для ускорения обучения и работы с данными размерность была понижена до 10% от общего размера датасета. Выведем для датафрейма метод .info().

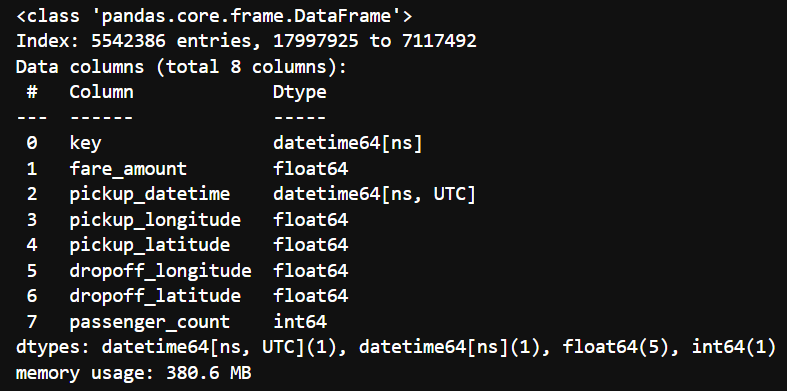


Рисунок 1 – Результат работы df.info

Следующий этап — это очистка данных. В рамках очистки данных из датасета были удалены пустые значения, который были найдены в двух колонках (dropoff\_longitude, dropoff\_latitude). Удаление не является оптимальной опцией при работе с данными, однако учитывая, что данные переменные невозможно предсказать, а также, что количество пропусков составляет менее 1% от общей размерности датафрейма (рис. 2, было принято решение удалить строчки с пропущенными данными.

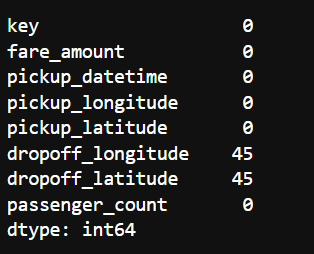


Рисунок 2 – Количество пустых значений в разрезе колонок

После очистки датасета от пропусков приступим к работе над выбросами. В нашем датасете имеется две колонки в которых возможны выбросы, это «passenger\_count» и «fare\_amount». Начнем с «passenger\_count», с помощью value\_counts определим количество значений и для удобства переведем их в проценты по отношению к общему размеру датасета.

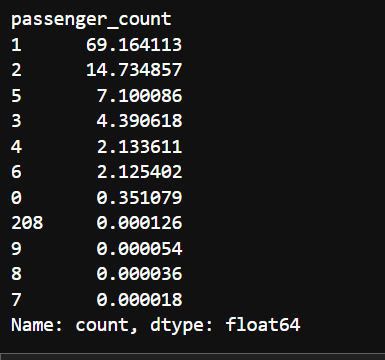


Рисунок 3 – Процент выбросов в переменной «passenger\_count»

Как можно видеть из рисунка 3 в переменной, которая описывает количество пассажиров имеется экстремальные значения составляющее минимальное количество в датасете, это все, что больше 6 пассажиров и минимальное значение пассажиров 0, что возможно является ошибкой. Так как численность выбросов менее 1% относительно выборки их можно удалить.

Также была проведена проверка на дублирующиеся строчки, в частности на одинаковые ID в колонке «key». По итогам проверки были удалены дублирующиеся строчки размером менее 1% от всего датафрейма.

С помощью .describe() посмотрим на переменную «fare\_amount».

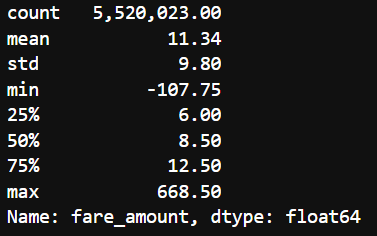


Рисунок 4 - Метод.describe() для переменной «fare\_amount»

Анализируя рисунок 4 можно увидеть, что минимальная цена составляет отрицательное значение, что обозначает ошибку в данных, а также анализируя информацию из открытых источников можно установить, что минимальная цена на такси в Нью-Йорке составляет 3$, общее количество значений меньше 3$ составляет менее 1% от общего размера датасета и их удаление не повлияет на качество обучение, следовательно удалим отрицательные значения, так как они ошибочны.

Так как Нью-Йорк находится неподалеку от водных объектов вероятны ошибки в местоположении точки отправления или точки конца поездки для того, чтобы определить наличие ошибок воспользуемся библиотекой Folium и картой heatmap для визуализации распределения заказов в Нью-Йорке.

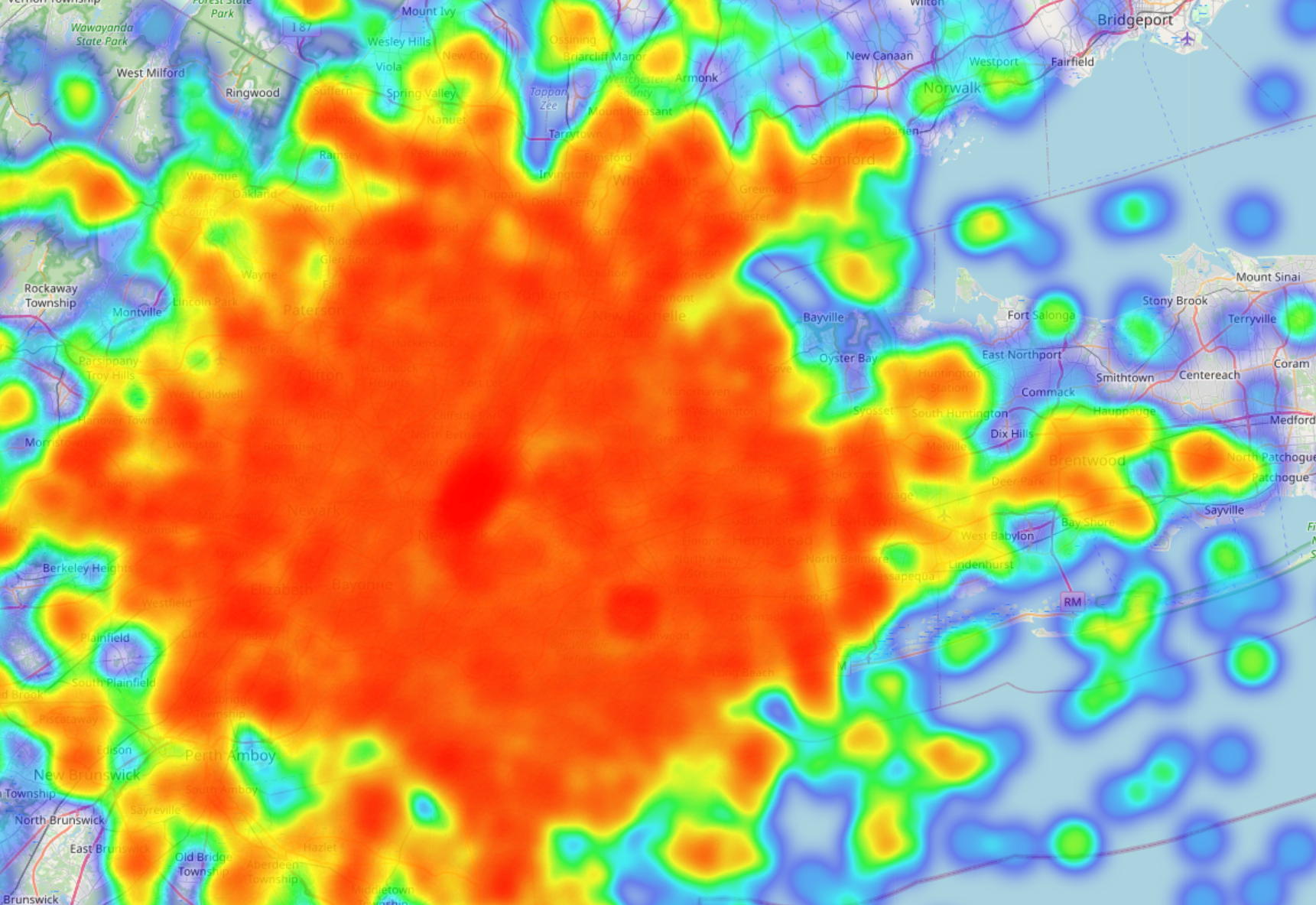


Рисунок 5 – Тепловая карта распределения заказов такси

На рисунке 5 можно увидеть, что существуют точки, находящиеся на территории водных объектов, что указывает на наличие ошибок в градусных мерах. Очистим датасет от точек находящихся на территории водных объектов с помощью датасета о границах районов Нью-Йорка.

После очистки датасета приступим к формированию новых признаков из уже имеющихся. Для формирования цены на такси очень важно расстояние от точки старта маршрута до точки его завершения. С помощью формулы Хаверсайна (рис. 6) можно определить расстояние в километрах от точки до точки с помощью широты и долготы. Создадим новый признак «distance\_km», который будет составлять длину маршрута в километрах.

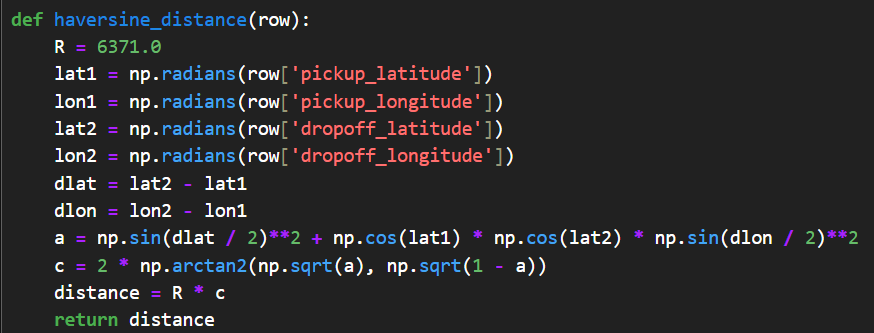


Рисунок 6 – Функция для определения длины маршрута

Также для нашего исследования вычленим из признака «pickup\_datetime» час, день и месяц начала поездки в отдельные признаки. А также для того, чтобы модель лучше понимала признаки времени приведем их к формату циклического кодирования, когда переменная представляет из себя двумерный вектор, где один компонент – синус, второй – косинус.

Для формирования оптимального прогноза также создадим новые переменные с данными о районе посадки и высадки. Информацию о районах и их координатах получим с помощью открытого датасета «Borough\_Boundaries\_20250608».

Приступим к следующему этапу формирования модели – моделированию. Для моделирования, в целях экономия времени размерность также была понижена до 10% от уже имеющейся.

Для первой модели возьмем ElasticNet() из пакета sklearn.linear\_model. В качестве основных метрик возьмем RMSE и коэффициент детерминации.

Результаты обучения модели можно увидеть на рисунке 7.



Рисунок 7 – Метрики для ElasticNet()

Первое значение – метрика RMSE, второе – коэффициент детерминации. Метрики модели имеют средний показатель.

После преобразования обучим несколько моделей. Результаты обучения представлены в таблице 1.

Таблица 1 – Результаты работы моделей

|  |  |  |  |
| --- | --- | --- | --- |
| Номер модели | Краткое описание модели | Качество модели на кросс-валидации | Качество модели на leaderboard |
| 1 | ElasticNet( | 4,10523 | 5,50735 |
| 2 | DecisionTreeRegressor( | 3,85453 | 4,24815 |
| 3 | RandomForestRegressor( | 3,61208 | 4,17761 |
| 4 | GradientBoostingRegressor( | 3,60127 | 4,20229 |
| 5 | XGBRegressor( | 3,63924 | 4,17037 |
| 6 | KNeighborsRegressor( | 3,85824 | 4,52623 |

Анализируя таблицу 1 видно, что лучший результат показала XGBRegressor.

Модели требуют улучшения, этого можно добиться с помощью увеличения количества признаков, описывающих целевую переменную, например это может быть сочетание признаков часа суток в будний день и района начала поездки. Также можно воспользоваться опцией более глубокого изучения переменных и более подробного анализа взаимосвязей между ними. Также для более качественного обучения моделей можно воспользоваться более детальной настройкой гиперпараметров для моделей машинного обучения, а также можно использовать другие модели. Для улучшения качества модели стоит воспользоваться методом главных компонент для определения влияния признаков на целевую переменную.