NYC Trip Duration Prediction Report

Table of Contents

1. About the Report

In the NYC trip duration prediction report, I will present the full process I went through to achieve high performance and improve the model step by step. This report shows how **feature engineering** and a strong understanding of the domain can greatly impact model performance, and **how Exploratory Data Analysis (EDA)** helps deepen your thinking about the problem and improve your understanding of it. especially in a classic machine learning problem.

The report is written in detail, so it can also serve as a reference for anyone who wants to understand this challenge.

Note: The core of the report is **Section 4 (Baselines & Improvements)**, which includes all details. If you already have prior knowledge about the problem, you can jump directly to that section to save time.

2. Problem Definition

The problem comes from a Kaggle competition, and the objective is to build a model that predicts the total ride duration of taxi trips in New York City.

3. Dataset Overview

The primary dataset is one released by the NYC Taxi and Limousine Commission, which includes pickup time, geo-coordinates, number of passengers, and several other variables.

Main Features:

- **id** - a unique identifier for each trip
- **vendor_id** - a code indicating the provider associated with the trip record
- **pickup_datetime** - date and time when the meter was engaged
- **dropoff_datetime** - date and time when the meter was disengaged
- **passenger_count** - the number of passengers in the vehicle (driver entered value)
- **pickup_longitude** - the longitude where the meter was engaged
- **pickup_latitude** - the latitude where the meter was engaged
- **dropoff_longitude** - the longitude where the meter was disengaged
- **dropoff_latitude** - the latitude where the meter was disengaged
- **store_and_fwd_flag** - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
- **trip_duration** - duration of the trip in seconds

Engineered Features:

In **Section 4 (Baselines & Improvements)**, you will find all the engineered features I applied, but these are the most important ones:

Haversine Distance, **Manhattan Distance**, **Log-transformed Trip Duration, Log-transformed Haversine Distance, and Time Features**

4. Baselines & Improvements

Note: The baseline does not represent the final model. It simply reflects my initial thought process and the steps I followed to reach the final model. In the end, I will present two final models.

In this section, we evaluate several baseline models for predicting trip duration, starting with simple approaches and iteratively improving performance through feature engineering, outlier handling, and model refinements. Each baseline builds on the previous one, incorporating lessons from exploratory data analysis (EDA) to enhance predictive accuracy. We use Ridge regression (with alpha = 1) as the modeling algorithm across all baselines due to its effectiveness in handling multicollinearity and providing stable predictions for regression tasks. Performance is measured using Mean Squared Error (MSE) and R^2 score on a held-out validation set.

general note :

Some features contain outliers, and there are many ways to handle them. In this project, I will apply three approaches:

1. Replace the outliers with NaN and then drop all NaN values.

2. Clip the outliers.

3. keep them

I will report the best results obtained from these methods. All the performance numbers presented in each baseline will be based on the best-performing approach, and I will clearly indicate which method was used at the start of each baseline.

Note: In all baselines, **i discover that keeping the outliers give the smae performance** or better thant clipping or removing . (Added after completing the experiments.)

so removing or clipping the outlier is useless

4.1 Baseline 1 :

For the first baseline, we started with the original dataset and applied basic feature engineering to enhance the predictive power of the input features. The raw features included:

id: A unique identifier for each trip (dropped as it provides no predictive value).

vendor_id: A code indicating the provider associated with the trip record (dropped due to low correlation with trip duration).

pickup_datetime: Date and time when the meter was engaged (used to extract temporal features).

passenger_count: The number of passengers in the vehicle (dropped after EDA showed minimal impact on duration).

pickup_longitude and **pickup_latitude**: Longitude and latitude where the meter was engaged (used for distance calculation).

dropoff_longitude and **dropoff_latitude**: Longitude and latitude where the meter was disengaged (used for distance calculation).

store_and_fwd_flag: Flag indicating if the trip record was stored in vehicle memory before sending (dropped as it had negligible correlation with the target).

trip_duration: Duration of the trip in seconds (target variable, transformed for modeling).

Feature Engineering Steps

To improve model input, we performed the following transformations:

Temporal Feature Extraction

- Converted **`pickup_datetime`** into a proper datetime object.
- Extracted key time-based features:
    - **start_hour**: hour of the day (0–23)
    - **day_of_week**: day index (0 = Monday, …, 6 = Sunday)
    - **weekend_day**: binary flag (1 if Saturday/Sunday, else 0)
    - **month**: calendar month (1–12)
- Created a categorical feature **which_part_of_day** based on **`start_hour`**:
    - Morning: 5 AM – 12 PM
    - Afternoon: 12 PM – 5 PM
    - Evening: 5 PM – 9 PM
    - Night: 9 PM – 5 AM

Note: none of these features has a great effect on performance

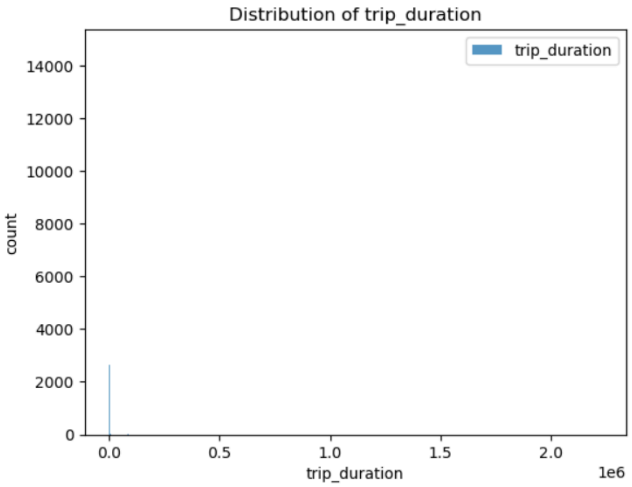
2 - **Distance Calculation**:

- Computed the Haversine distance (in kilometers) between pickup and dropoff coordinates using the formula:

$$d = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

- This feature captures the straight-line distance, which is **a strong proxy** for trip duration.

3 - Target Variable Analysis



The raw trip duration distribution reveals some notes :

1 -Raw Distribution of Trip Duration

Extreme Right Skewness: Most trips are concentrated near zero, but the tail extends to ~2227612 seconds (~23 days).

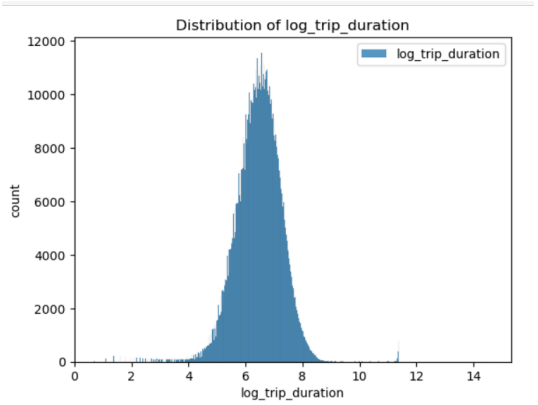
Unrealistic Outliers: Trips shorter than 60 seconds or longer than 6–24 hours are impossible and likely caused by data entry errors or faulty meters.

Statistical Concerns: The mean and standard deviation are inflated, and the raw distribution violates linear regression assumptions.

Modeling Impact: Training on this skewed distribution would make the model focus too much on extreme trips and reduce its ability to learn from normal trips.

Key observation: Preprocessing is essential, including filtering unrealistic values and applying transformation techniques.

2- Log-Transformed Distribution of Trip Duration



Bell-Shaped Distribution: Transformed from extreme skewness to a near-normal distribution

Key Insights

1. **Most trips cluster between log values 6–7**
    - Which means the **trip duration is between 6.7 minutes and almost 20 minutes**.
    - This is the **typical taxi trip duration and more logical for taxi trip duration** .
2. **Peak around log ≈ 6.5**
    - log(6.5) ≈ 665 seconds ≈ 11 minutes.
    - inform that the most common trip duration is **10–12 minutes**.
3. **Long right tail**
    - The distribution extends beyond log = 8 (50 minutes).
    - These are potential **outliers**

Transformation Benefits:

Bell-Shaped Distribution: Transformed from extreme skewness to a near-normal distribution

Statistical Suitability: Distribution now suitable for linear regression and ML models

Outlier Detection: Easier identification of remaining extreme values(Detecting outliers more easily)

Model Performance: Improved training stability and prediction accuracy

Metrics	without applying a log transformation to the trip duration feature	with applying a log transformation to the trip duration feature
R2 Score(train)	0.0152	0.3262
R2 Score(val)	0.0036	0.3664
MSE(train)	14841555.2791	0.4256
MSE(val)	82501285.5994	0.4055

The outlier range was defined as trips shorter than 150 seconds or longer than 22,000 seconds

Impact of Clipping, removing, and keeping Outliers on Performance

Metrics	applying clipping	applying outlier removal	Keeping the outliers
R2 Score(train)	0.3642	0.3800	0.3262
R2 Score(val)	0.3616	0.3487	0.3664
MSE(train)	0.3377	0.2844	0.4256
MSE(val)	0.4086	0.4168	0.4055

From the table, **clipping or removing the outliers is not particularly important or critica**l, as the results are quite similar.

Baseline 1 result (The best results were obtained when keeping the outliers.):

Metrics	Train	Val
R2 Score	0.3262	0.3664
MSE	0.4256	0.4055

4.2 Baseline 2 :

While Baseline 1 focused on leveraging the original dataset with basic temporal features, in Baseline 2, I expanded the feature set by introducing and **manhattan_distance and more investigation on distance features.** These enhancements were designed to improve the model's ability to predict trip duration**. To maintain consistency, I** used **Ridge regression** (alpha = 1).

Feature Engineering Additions

In addition to the features from Baseline 1 (start_hour, day_of_week, weekend_day, month, which_part_of_day dummies, and distance), we incorporated:

1- manhattan_distance (CITY BLOCK DISTANCE)

In the beginning, I needed to calculate the distance as a proxy for trip duration. At first, I only knew about the haversine distance and was not aware of the Manhattan distance. After doing some research, I found that adding more distance-related features could help the model capture better patterns and improve its predictive ability.

The key differences between the two are:

- **Haversine distance: the shortest straight-line distance between two points (ignores roads).**
- **Manhattan distance: the distance measured along a grid of streets, which is closer to real taxi travel.**

Therefore, I added Manhattan distance to this baseline to see its impact on performance and to check whether it is more suitable for this problem. Let's evaluate this through the results.

2.1 Effect of including the Manhattan distance feature on model performance

Note: These results are based on the Baseline 1 features with keeping trip duration outliers, in addition to the newly added feature.

Metrics	without (manhattan_distance) feature	with (manhattan_distance) feature
R2 Score(train)	0.3262	0.3242
R2 Score(val)	0.3664	0.3664
MSE(train)	0.4256	0.4268
MSE(val)	0.4055	0.4074

observations :

Impact of Adding Manhattan Distance

- The inclusion of the Manhattan distance feature did not have any noticeable impact on the results. It might lead to a slight improvement in future experiments, or it may continue to have no significant effect.

Comparison Between Manhattan and Haversine Distance

- I also tested the effect of removing each feature separately.

    a. Removing the Haversine distance feature and replacing it with Manhattan distance led to a performance drop of approximately 2%..

    b. On the other hand, removing the Manhattan distance feature had no noticeable impact on performance..

Conclusion :

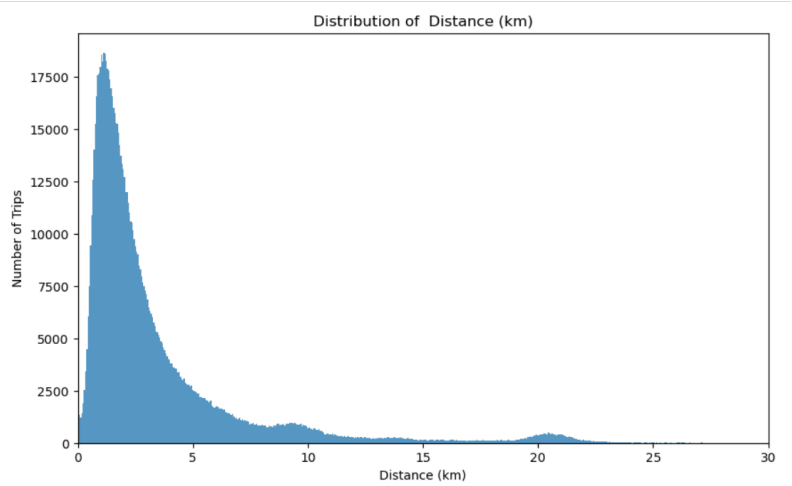
Using both distance features together may slightly improve performance, but the increase is expected to be minimal. However, if you choose to use only one, I recommend using the Haversine distance feature.

3 - Further Investigation of the Haversine Distance Feature

In **Baseline 1**, I had already added the ***haversine_distance*** feature. However, I did not analyze it earlier because my main focus was on **the *trip_duration*** feature. In this baseline, I now analyze ***haversine_distance*** to see what insights it provides.

3.1 Analysis of haversine_distance Distribution

The histogram shows the distribution of **trip distances (in km)** across the dataset.



Note: For better visualization, I plotted the complete graph but limited the x-axis to 30 using xlim(0,30).

1- Right-Skewed Distribution:

- Most trips are **short-distance**, concentrated between **0.5 km and 5 km**, with a strong peak around **1–2 km**.
- This makes sense in the context of NYC taxi trips, as most rides are relatively short.

2- **Long Tail of Longer Trips:**

- The distribution has a long right tail, with trips extending beyond **25 km**.
- These extreme values are likely outliers

3.2 Effect of dropping and clipping distance outliers on the model's performance

After analyzing the distribution of **haversine_distance**, I noticed there are outliers. To evaluate their impact, I removed and clipped these outliers and observed the effect on model performance. The results are presented in the next table.

Note: I kept only trips with distances in the range of 1 to 30 km

Metrics	dropping outliers from distance and trip duration together	clipping outliers from distance and trip duration together	dropping outliers only from the distance feature	clipping outliers only from the distance feature
R2Score(train	0.4894	0.4467	0.4498	0.3956
R2 Score(val)	0.3523	0.3926	0.3530	0.3935
MSE(train)	0.1968	0.2939	0.2355	0.3817
MSE(val)	0.4145	0.3887	0.4141	0.3882

observation about dropping distance outliers

From the table above, I observed a noticeable improvement in model performance after filtering trips based on the valid distance range (1–30 km). To further investigate, I experimented with different distance thresholds:

- Range: 1–30 km
    - Model performance improved compared to using the full dataset.
- Range: 10–30 km
    - The model achieved an extremely low R2 score, reaching a negative score.

In general, I noticed that as the lower bound of the distance range increased, the model performance kept decreasing until it reached negative values.

Interpretation :

1. **Most taxi trips in NYC fall within the 1–5 km range**, which means that filtering out distances below 10 km removes the majority of real-world trips.

2. This indicates that the model was trained on fewer and less diverse trips. When evaluated on the validation set**, where most trips fall within the 1–5 km range**, the R^2 score dropped to **negative values.** This makes it easier to obtain an unrealistically low R^2, which is expected and makes sense given the data distribution.

observation about clipping distance outliers

Clipping the lower bound **did not show a fixed pattern**: with some values, performance improved, while with others, it slightly decreased. However, the decrease was minor and **never resulted in negative R^2 values** as observed when outliers were dropped.

As for the upper bound, increasing it to 50 caused only a minor change in performance, which can be safely ignored.

4 - Baseline 2 Final results

In this baseline, I will not present the final results because there is another aspect I plan to explore separately. I will create a new baseline to evaluate whether polynomial regression or polynomial features have a significant impact on the model's performance.

5 - Key Insights Summary (Baseline 2)

1. Adding the **Manhattan distance** feature **had little to no impact** on the model's performance.

2. Removing outliers from **haversine_distance** had no noticeable impact compared to the final results of **Baseline 1**; in fact, performance decreased by about 1% in both cases of feature removal, as mentioned in the table above. Overall, it's **worth trying** for discovery purposes.

3. Clipping outliers from **haversine_distance** improved the R^2 score by about 3% in both cases of feature clipping, as mentioned in the table above, compared to the final results of **Baseline 1**. It's a better approach since it preserves more data for training rather than discarding it. However, the improvement is relatively small, and the gap between the training and validation R^2 scores remains noticeably large. Therefore, while it was worth trying for exploration purposes, I will continue without clipping or removing outliers.

4.3 Baseline 3 :

In this baseline, no additional features were introduced; instead, Polynomial Feature Transformation was applied to certain features in an attempt to enhance the model's performance. Applying polynomial feature transformation allowed the model to capture non-linear relationships between certain variables and the target.,

1- Selected features for applying polynomial regression

- The best performance improvements were achieved when applying polynomial Feature transformation to **haversine_distance**, **start_hour, and manhattan_distance**.

- Among these features, **haversine_distance** made the most significant contribution to the model's performance, while **start_hour** and **manhattan_distance** had little to no impact.
- Additionally, replacing **haversine_distance** with **manhattan_distance** resulted in almost the same performance, indicating that both distance measures capture similar information in this context.

2- The impact of applying polynomial feature transformation on the model performance (degree 2 vs degree 3)

To evaluate the effect of polynomial expansion, experiments were conducted using degree 2 and degree 3 transformations on ***haversine_distance*, start_hour, and Manhattan distance.**

Note: In this baseline, we only a**pplied outlier removal or clipping to the trip_duration** feature, and the results are based on that.

Metrics	degree 2 (clipping)	degree 2 (dropping)	degree 2 (without clipping or dropping)	degree 3 (clipping)	degree 3 (dropping)	degree 3 (without clipping or dropping)
R2Score(train	0.5599	0.4347	0.5012	0.5975	0.4370	0.5378
R2 Score(val)	0.4993	0.3664	0.5008	0.5331	0.3674	0.5350
MSE(train)	0.2338	0.2593	0.3151	0.2138	0.2583	0.2919
MSE(val)	0.3205	0.4055	0.3195	0.2989	0.4049	0.2976

observation :

- The degree 3 model achieved a higher R^2 score.However, this improvement came with a significant **increase in complexity** due to the larger number of interaction terms.

- A degree 3 model may **memorize** the training data instead of **generalizing**, which goes against the main goal of machine learning to **generalize well to unseen data.**

- **Degree 3** polynomials greatly increase the number of features, which leads to longer training time and higher memory usage. If applied to additional features beyond distance and start_hour, the feature space expands explosively.

- The **degree 2 model**, in contrast, delivered strong predictive performance. It remained more interpretable, computationally efficient, and less prone to overfitting.

- When **removing outliers from both the trip_duration and haversine_distance feature**s and applying a polynomial degree of 3, the R^2 score **drops to negative values.**

so finally I choose the degree 2 polynomial regression, as it achieve a better balance between accuracy, interpretability, and robustness.

4.4 Baseline 4 :

In this baseline, we continue from **Baseline 2**, which achieved an **R² score of 0.3616** on the validation dataset. Looking at the results of **Baseline 3**, one might assume it is the best model and ready for deployment. However, the key question is: **can we achieve the same results without polynomial feature transformation**, or is Baseline 3 truly the best model to deploy?

1- Feature Engineering Additions

1.1 Log Transformation of Haversine Distance

As mentioned previously in Baseline 2, the **haversine_distance distribution is right-skewed**, so we can apply **a log transformation** to make it closer to a normal distribution, which may help the linear model and improve performance. Let's apply this transformation and see whether the impact is significant enough to achieve an R² score close to or above Baseline 3, or if Baseline 3 remains the best model for deployment.

Metrics	Baseline 2	Baseline 4
R2 Score(train)	0.3262	0.6027
R2 Score(val)	0.3664	0.6040
MSE(train)	0.4256	0.2509
MSE(val)	0.4055	0.2534

observation :

- You can clearly see the strong impact on performance just from applying a log transformation, without adding any new features. This raises an important question: **is the polynomial feature transformation applied in Baseline 3 really worth it?** Let's look at the results in the table, and I will highlight the reasons why I prefer this transformation over polynomial feature transformation.

1.2 Baseline 3 vs. Baseline 4: Is Polynomial Transformation Worth It?

Metrics	Baseline 3	Baseline 4
R2 Score(train)	0.5012	0.6027
R2 Score(val)	0.5008	0.6040
MSE(train)	0.3151	0.2509
MSE(val)	0.3195	0.2534

According to the above table, here are some observations:

Although Baseline 3 shows a significant performance improvement compared to Baseline 2, and its results are close to Baseline 4 compared to the other baselines, I chose the **Baseline 4 model** for the following reasons:

1. **Baseline 3 increases model complexity** by adding polynomial features.

2. This added complexity introduces a **higher risk of overfitting**, especially if applied to more features.

3. **Baseline 4 does not increase complexity** or introduce new features. Instead, it simply applies a **log transformation** to the haversine distance, which is right-skewed. This makes the distribution more normal and helps linear models perform better.

4. Log transformation is also **easier to interpret.**

5. It provides **better generalization** potential since it's a simple feature transformation, not an artificial expansion of the feature space.

Note: Features such as (**vendor_id, passenger_count, store_and_fwd_flag)** have no noticeable impact on model performance.

4.5 final model features (configuration) :

1- **Temporal Feature Extraction**

- **start_hour**: hour of the day (0–23)
- **day_of_week**: day index (0 = Monday, …, 6 = Sunday)
- **weekend_day**: binary flag (1 if Saturday/Sunday, else 0)
- **month**: calendar month (1–12)
- Created a categorical feature **which_part_of_day**

2- Distance feature: haversine_distance

3- Log transformation: applied to trip_duration and haversine_distance

4 - Outliers: no clipping or removal applied to any features

5- Polynomial transformation: not applied

5- Conclusion

1- Clipping or removing outliers in this project is useless, as it has no positive impact and, in some cases, even harms performance.

2- Conducting EDA is essential to understand the distribution and behavior of the features.

3- Exploring other models in future work could provide further improvements.

4- Applying polynomial features in this project did not have a significant impact as expected theoretically, though it may be more beneficial in other contexts.

6- final model correlation matrix