Data Science Project

Automatically task solving dominance by Manus AI

Tobia Ippolito

Contents

1 Proceeding

After hearing about the project I was directly sure which AI would be dominant and state-of-the-art in solving a whole task - including multiple steps like data loading, cleaning, feature engineering, ... The AI Chatbot I used is ManusAI to solve the Kaggle Competition "New York City Taxi Fare Prediction" [1][2]. This AI is meant to solve more complicated tasks with multiple steps and is more meant as a real AI worker which creates a to do list for himself and sophistically validates his own work by running his created code to check if it really works.
To get the best result possible I used ChatGPT (with GPT 4 as background model) to generate an fitting prompt for Manus AI, which probably was not needed but still used. I asked:

> Generate me an command/prompt for an AI Chatbot to solve https://www.kaggle.com/competitions/new-york-city-taxi-fare-prediction in a google colab notebook (ipynb), I want best results.Task: New York City Taxi Fare Prediction => Can you predict a rider's taxi fare?
>
> With optional Data Analysis and Feature Engennering

The answer was:

> You are an expert data scientist.
> Please generate a full Google Colab notebook (as Python code in .ipynb format) that solves the Kaggle competition New York City Taxi Fare Prediction with high accuracy.
>
> Include the following:
>
> - Kaggle API setup to download the data
> - Exploratory Data Analysis (EDA) with visualizations
> - Data cleaning and preprocessing
> - Advanced feature engineering (e.g., datetime features, Haversine distance, clustering, etc.)

- Model training using XGBoost, LightGBM, and optionally CatBoost
- Model evaluation with RMSE
- Hyperparameter tuning (optional, but preferred)
- Prediction on the test set
- Save and export a submission.csv file ready for Kaggle submission

Aim for state-of-the-art performance while keeping code modular and clean. Include inline comments and explanations in markdown cells to guide understanding. Assume the user may only run the notebook in free-tier Google Colab, so subsample the training data as needed.
The goal is to maximize leaderboard score while remaining reproducible.

With this prompt I generated the solution in ManusAI which directly gave me SOTA results and analytic insights in the named kaggle competition.

Unlike ChatGPT or other chatbots the generation needed about 11 minutes and ManusAI gave himself following to do's:

Competition Details

Goal: Predict the `fare_amount` for taxi rides in New York City.
Evaluation Metric: Root Mean Squared Error (RMSE).
Data Files:

- `train.csv`: Training data (approx. 55M rows).
- `test.csv`: Test data (approx. 10K rows).
- `sample_submission.csv`: Sample submission file.

Key Features:

- `key`: Unique identifier for each row.
- `pickup_datetime`: Timestamp of the pickup.
- `pickup_longitude`, `pickup_latitude`: Coordinates of the pickup location.
- `dropoff_longitude`, `dropoff_latitude`: Coordinates of the dropoff location.
- `passenger_count`: Number of passengers.

Submission File Format:

`key,fare_amount`

Plan for Notebook Generation

1. **Kaggle API Setup:** Instructions for setting up Kaggle API credentials to download data.
2. **Data Loading and Subsampling:** Load `train.csv` and `test.csv`. Subsample `train.csv` for free-tier Colab compatibility.
3. **Exploratory Data Analysis (EDA):**
   - Summary statistics.
   - Distribution of `fare_amount` and `passenger_count`.
   - Geospatial analysis of pickup/dropoff locations.
   - Time-based analysis (`pickup_datetime`).
4. **Data Cleaning and Preprocessing:**
   - Handle missing values.
   - Remove outliers (e.g., fare amount, coordinates).
   - Address illogical values (e.g., `passenger_count` = 0).
5. **Advanced Feature Engineering:**
   - **Datetime Features:** Extract year, month, day, day of week, hour from `pickup_datetime`.
   - **Haversine Distance:** Calculate distance between pickup and dropoff coordinates.
   - **Geospatial Features:** Consider features like distance to NYC landmarks (JFK, EWR, LGA airports, Manhattan, etc.), or create clusters of pickup/dropoff locations.
   - **Fare per Mile/Minute:** (If applicable after initial feature engineering).

6. **Model Training:**
   - Implement XGBoost and LightGBM models.
   - Briefly explain model choices.
7. **Model Evaluation:**
   - Use RMSE as the primary evaluation metric.
   - Cross-validation strategy.
8. **Hyperparameter Tuning:**
   - Basic tuning for selected models (e.g., using `GridSearchCV` or `RandomizedSearchCV` on a subsample).
9. **Prediction and Submission:**
   - Predict `fare_amount` on the test set.
   - Generate `submission.csv` in the required format.

This detailed plan will guide the notebook creation process, ensuring all requirements are met and the notebook is well-structured and reproducible.

The generated python notebook is well structured and on point, a professional could not do better and most likely much worse in 11 minutes. The notebook is provided separately next to this writing.
The created notebook includes:

1. Downloading and importing needed external libraries
2. Downloading the 'new-york-city-taxi-fare-prediction' dataset
3. Sampling 50k training data and all 9914 test data (only input is available in test)
4. Analyzing with general data statistics, fare amount distribution, passenger count distribution, geospatial analysis and time-based analysis
5. Data Cleaning and preprocessing -> remove null values, outliers and unlogical values
6. Feature Engineering -> hour, day of the week, month, year, day of the year, week of the year, haversine distance, manhatten distance and distances to important places (airport, midtown, ...)
7. Model Training -> training XGBoost and LightGBM with RSME as error metric as described in the competition
8. Prediction in the right submission format

So it needed only one prompt to generate a complex and already validated and working code.

2 Results

The notebook already applies the right format to submission the test data result on the kaggle website. Figure 2.1 shows the result of both models which is not SOTA but would take the 540th place for the XGBoost model and for the LightGBM model the 548th place on the leaderboard, which means the models are in the top 36% and that with 11 minutes generation and with no fine-tuning from features or hyper-parameters and no direct data scientist help.

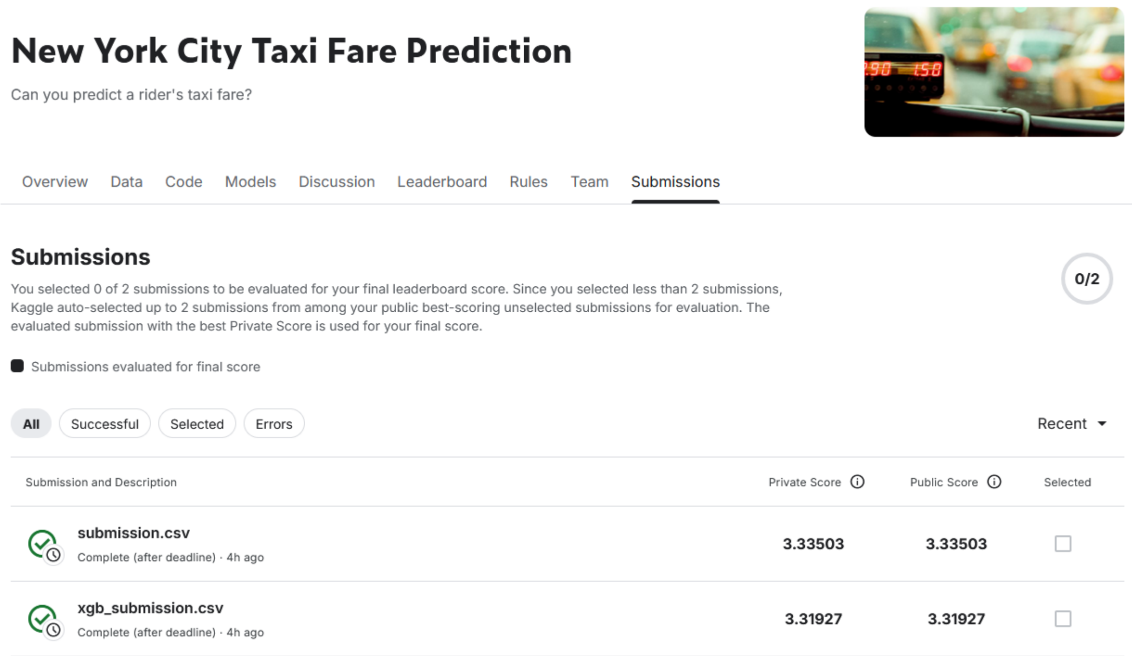

Figure 2.1: Results on the New York City Taxi Fare Prediction Kaggle Competition. 'submission.csv' is the result of the trained LightGBM model and 'xgb_submission.csv' is the result of the XGBoost model.

Next it would be interesting how much AI agents like ManusAI could make the result better and with how much effort. Also interesting is the question, how other Chatbots and AI agents would end up on the leader board and with how much effort. Lastly an investigation in more unknown tasks would be interesting, my guess is that ManusAI would be superior in more unknown tasks in comparison to ChatGPT or other Chatbots and in such a common task like the Taxis Fare which is propably in the train data of todays AI agents/Chatbots the difference is not big between ManusAI and other Chatbots, I guess.

3 Error Cases / Difficulties

The generation process was almost perfect and without effort. Just the python notebook has some mistake in the format which I solved by copy and paste every cell in a new notebook. I guess it was a small issue because the raw code looked pretty good and I could not saw an mistake on one overview. This sounds as a bug which may is only a single/rare occurrence or a bug which is most likely fixed in the future.
The code itself worked smooth without any changes needed. Only the API token from kaggle was expected on a place where it not was, so I had to correct the path but the AI could not know that, so it should not be counted as a miss-function. And for downloading the dataset your kaggle account have to participate on the kaggle competition which the AI could not do but a hint that the human operator have to do so would have been helpful.

In summary ManusAI generated direct working and already tested code, which was easy to use with some issues with the output format.

Bibliography

[1] M. Shen and Q. Yang, *From mind to machine: The rise of manus ai as a fully autonomous digital agent*, 2025. arXiv: 2505.02024 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2505.02024.

[2] *Official website from manusai – the chinese chatbot for solving multisteps tasks*, https://manus.im/app, Accessed: 2025-06-10.