NYC Taxi Fare Prediction

Group 1

Guilherme, Alex, Rakshit, Seonhye, Michael

Agenda

1. Problem Background
2. Preprocessing
3. EDA
4. Models
5. GPU Acceleration
6. Results
7. Conclusion
8. Future Work
9. Utilized Class Topics

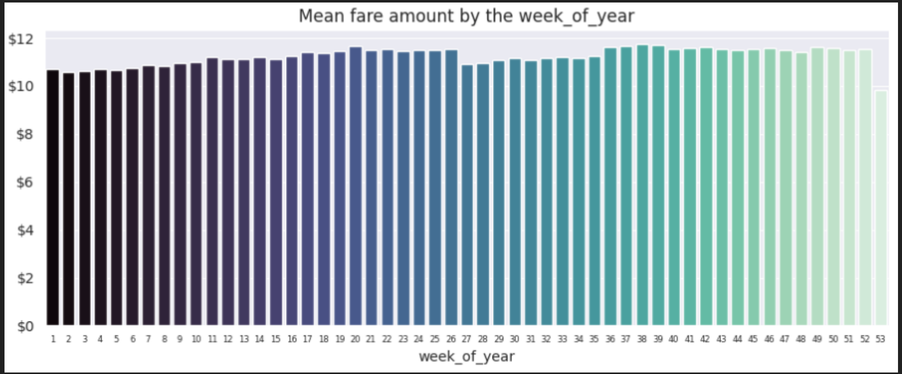
Problem Background

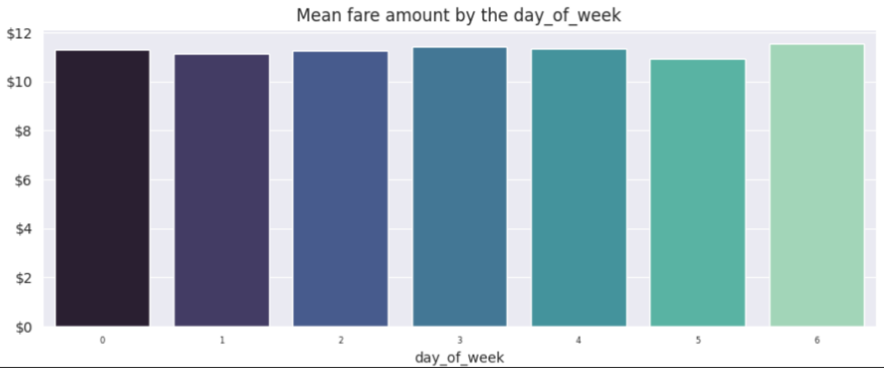
- **Target:**
  - Predict fare of a taxi ride
- **Features:**
  - Pickup time and coordinates
  - Drop Off coordinates
  - Passenger count
- **Historical Results:**
  - $3-5 MAE using just ride distance

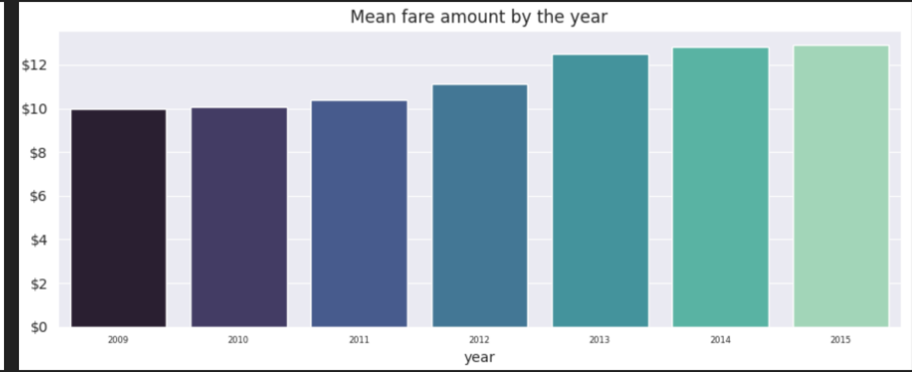
Preprocessing

- Drop rides that seem faulty/fake
  - 0 or more than 6 passengers
  - Below $3 and above $150
  - Low distance (< 1 km) with excessively high fares (>$40)
- Compute geographical distances
- Label pick-up and drop-off areas
  - Then one-hot encoded
- Dropped unused columns

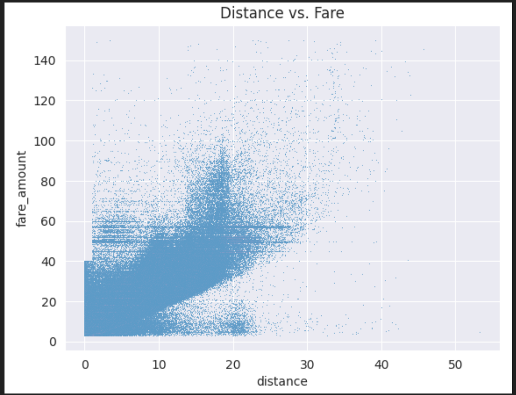
EDA - Features

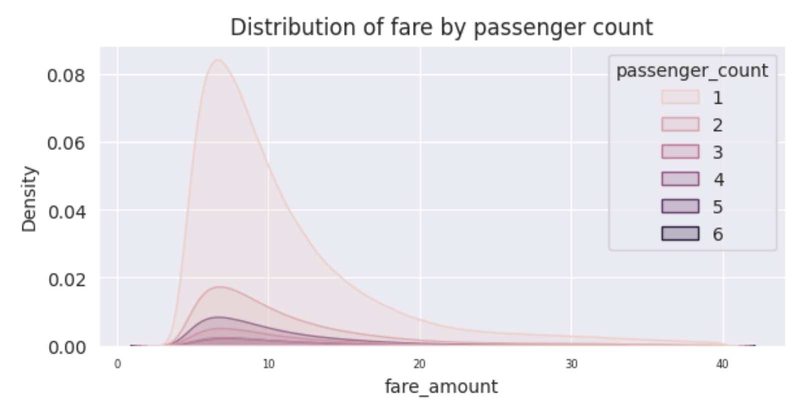


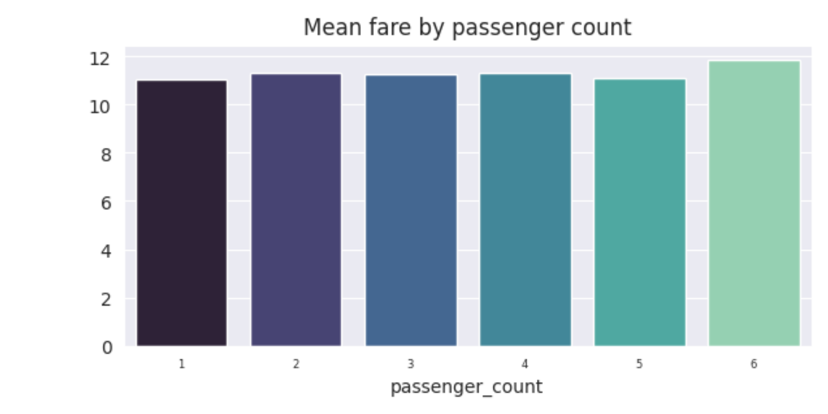




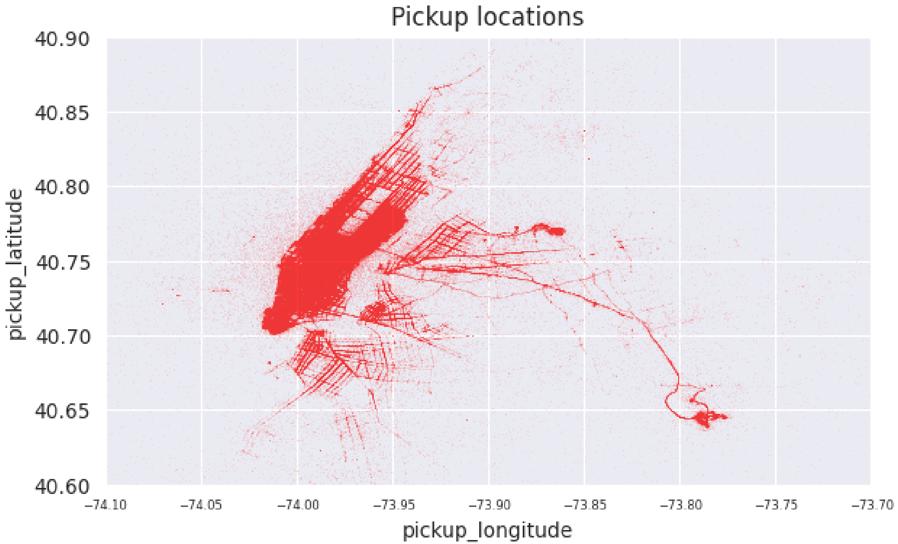
EDA - Features

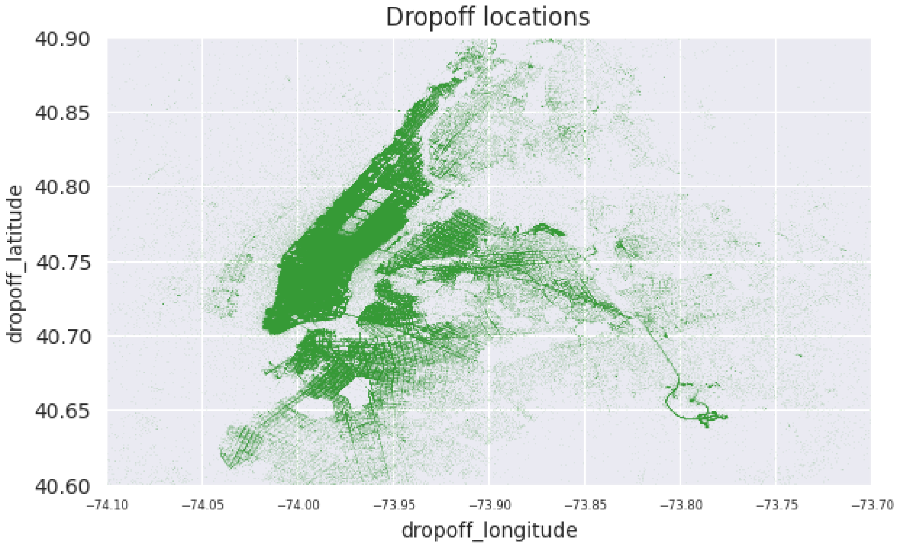




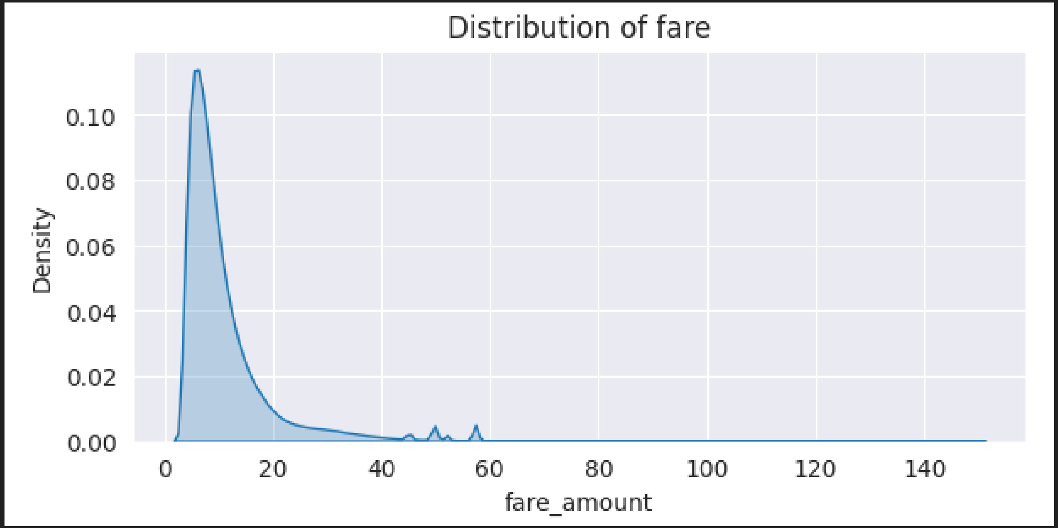


EDA - Features





EDA - Features



Models

- Linear Regression
  - Scikit-Learn and CuML versions
- Random Forest
  - Scikit-Learn and CuML versions
- XGBoost

❖ Model Objective: Minimize RMSE of Taxi Fare prediction

  ➢ RMSE penalizes larger errors

  ➢ We report Mean Absolute Error (MAE) for human understanding

GPU Acceleration

- We used CuDF and CuML libraries, developed by NVIDIA for work on NVIDIA GPUs.
- CuDF is a GPU-accelerated dataframe library that is API-compatible with Pandas
  - Utilizes similar Pandas syntax with GPU acceleration
- CuML is a machine learning library of machine learning algorithms optimized for GPU acceleration
  - Designed to work seamlessly with **CuDF**
- File size and device memory constrained how much data we could utilize
  - Required to use only 10% of given data to utilize **CuML**
  - Attempted PyTorch implementations failed entirely

Results

Device and Rows	Models	train_mae ($)	test_mae ($)	train_r2	test_r2	train_time (s)
CPU; 1 million rows	linear_regression	1.99	2.37	0.7742	0.7798	6.95
	random_forest	1.7	1.92	0.8361	0.8379	521.22
	xgboost	1.56	1.89	0.8683	0.8433	121.71
CPU; 5 million rows	linear_regression	1.99	2.36	0.7737	0.7793	30.51
	random_forest	1.71	1.92	0.8326	0.8377	3028.52
	xgboost	1.63	1.86	0.8491	0.8471	135.56
GPU; 5 million rows	linear_regression	2.05	2.41	0.7412	0.7624	1.91
	random_forest	1.71	1.92	0.8308	0.8377	462.58
	xgboost	1.63	1.86	0.8497	0.8468	9.94
CPU; full data (55mil)	linear_regression	1.99	2.36	0.7731	0.779	268.63
	random_forest	1.72	1.92	0.8301	0.8373	41023.48
	xgboost	1.63	1.85	0.8475	0.8481	1326.78

Conclusions

- Best Model: XGBoost
  - Best MAEs and R^2 scores
  - Ran very efficiently compared to Random Forest
- Linear Model close performance, but faster
  - More suitable for tasks demanding speed
- GPU fastest training speed
  - Suitable for tensor-like data
  - Harder to get devices with required memory
- Best Test MAE: $1.85
  - Still large error for fair prediction, needs improvement
  - Insignificant MAE difference between 5mil and full, significant speed difference

Future Work

- Utilize machines with more memory to use the full dataset
  - Properly tune model with such hardware
- Parallelize the computation more with more apt machines
- Utilize Google Maps API to try GPS distances and traffic levels as features
  - Required funding for practical use
- More precise EDA and cleaning
  - Finely remove outliers that cannot exist
    - Like being below base-rate
  - Breakdown pick-up and drop-off locations more finely
    - Like how going to an airport entails additional fees beyond regular rates

Utilized Class Topics

- Shell
- Python Performance
- Optimization
- Parallel Programming
- Python for GPUs

ANY QUESTIONS?