NEGATIVE BRANE CELLS
CSE 151B: TEAM #5

Melina Dimitropoulou-Kapsogeorgou

Daniel Lee

Jessie Ouyang

Benjamin Xia

SUMMARY

1. Key Words

2. Introduction

3. Exploratory Data Analysis/Feature Engineering

- The dataset has some crazy outliers, feature engineering is necessary

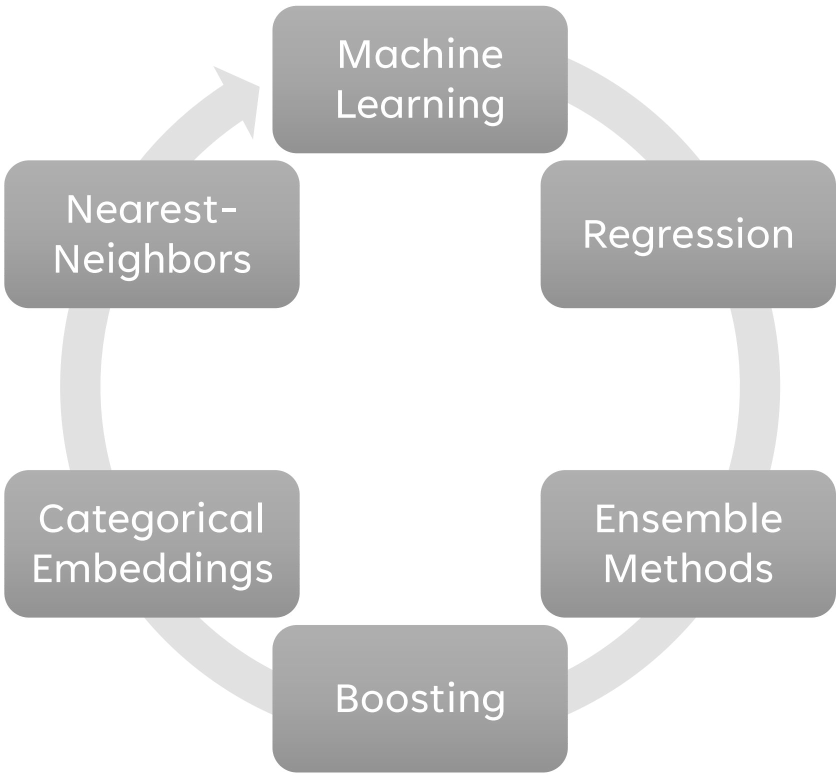
4. Our Approaches and Models

- We tried some MLP approaches after some basic models (linear regression, etc.)
- Ensemble methods seemed to perform better for this task.

5. Discussion and What We’ve Learned

- Importance of feature engineering to make models learn better and faster
- Effectiveness of different machine learning models in different applications

KEY WORDS



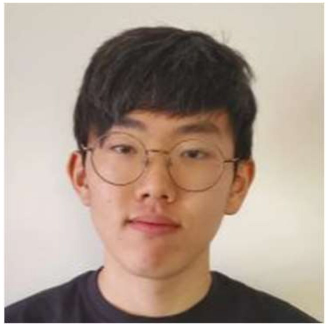
Machine Learning
Regression
Ensemble Methods
Boosting
Categorical Embeddings
Nearest-Neighbors

INTRODUCTION

TEAM INTRODUCTION

Melina

-31332 Brain Cells



Daniel

-42281 Brain Cells

Jessie

-78125 Brain Cells

Benjamin

-93211 Brain Cells

THE TASK

- Predict the travel time of taxi trips given certain metadata about each trip. (Regression)
- Solutions to this task could be applied for finding more optimal taxi or Uber scheduling/pairing clients with drivers.
- Use information such as timestamp, taxi ID, call type, origin call, origin stand, day type, etc. to predict travel times.

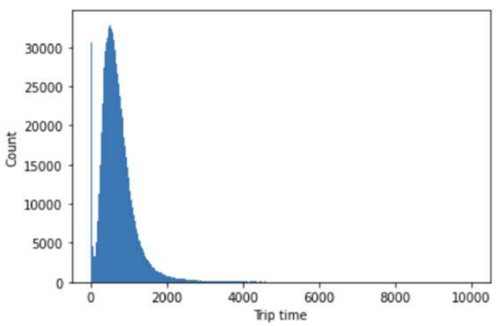
TRIP_ID	CALL_TYPE	ORIGIN_CALL	ORIGIN_STAND	TAXI_ID	TIMESTAMP	DAY_TYPE	MISSING_DATA
T1	B	NA	15	20000542	1408039037	A	FALSE
T2	B	NA	57	20000108	1408038611	A	FALSE
T3	B	NA	15	20000370	1408038568	A	FALSE
T4	B	NA	53	20000492	1408039090	A	FALSE
T5	B	NA	18	20000621	1408039177	A	FALSE
T6	A	42612	NA	20000607	1408037146	A	FALSE
T7	B	NA	15	20000310	1408038846	A	FALSE
T8	A	31780	NA	20000619	1408038948	A	FALSE
T9	B	NA	9	20000503	1408038563	A	FALSE
T10	B	NA	15	20000327	1408038021	A	FALSE
T11	B	NA	56	20000664	1408038267	A	FALSE
T12	C	NA	NA	20000160	1408038946	A	FALSE
T13	C	NA	NA	20000017	1408039130	A	FALSE
T14	C	NA	NA	20000312	1408036255	A	FALSE
T15	C	NA	NA	20000497	1408038388	A	FALSE
T16	C	NA	NA	20000440	1408037740	A	FALSE
T17	C	NA	NA	20000467	1408038804	A	FALSE
T18	C	NA	NA	20000338	1408038215	A	FALSE
T19	B	NA	15	20000101	1408038749	A	FALSE
T20	C	NA	NA	20000523	1408036754	A	FALSE
T21	B	NA	15	[illegible]	[illegible]	A	FALSE

METHODOLOGY

DATA ANALYSIS AND PREPROCESSING

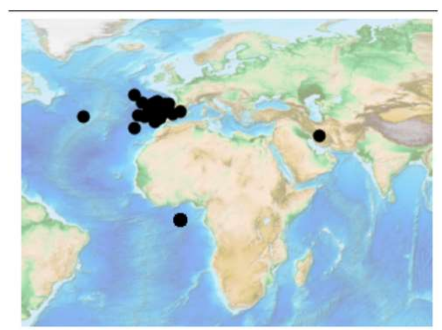
EXPLORATORY DATA ANALYSIS

Figure 1



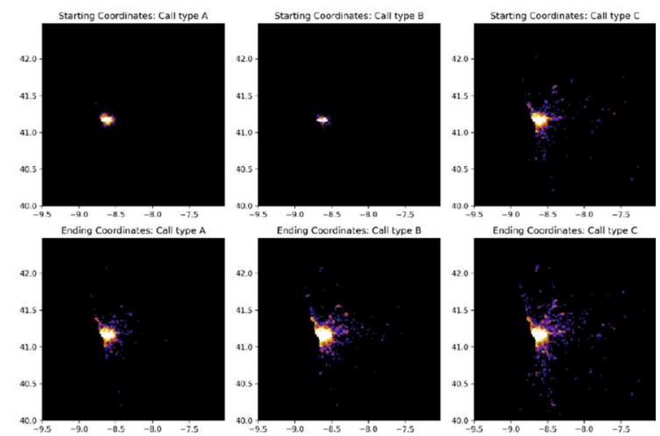
Training set follows a gamma or tweedie distribution

Figure 2



There are some crazy outliers

Figure 3



Call type A starting coordinates have a surprising amount of variance.

DATA PROCESSING

(NO SIGNIFICANT FEATURE ENGINEERING TRICKS YET)

1.Data Cleaning

- Getting rid of missing data entries
- Set NULL entries to 0.
- Getting rid of trip ID Column
- Getting rid of day type

2. Categorical

- Convert categorical data into a useful format
  - Call Type (One-hot encoding)
  - Taxi ID
  - Origin Call
  - Origin Stand

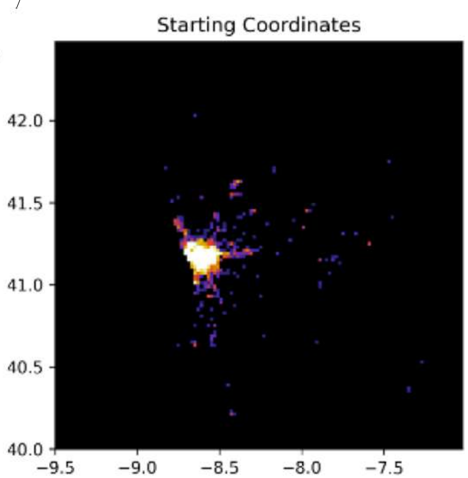
3. Time Encoding

- Split time into multiple features
  - Year
  - Month
  - Week of year
  - Day of week/month
  - Hour

4.Pruning

- Trip length > 30 seconds
- Trip length < threshold (varying)
  - $< \mu + 5\sigma$ (too small)
  - < 20000
  - < 15000 (best on public test)
  - Changed as project went on.
- Distance < 20km (city center)

ENGINEERING TRICKS

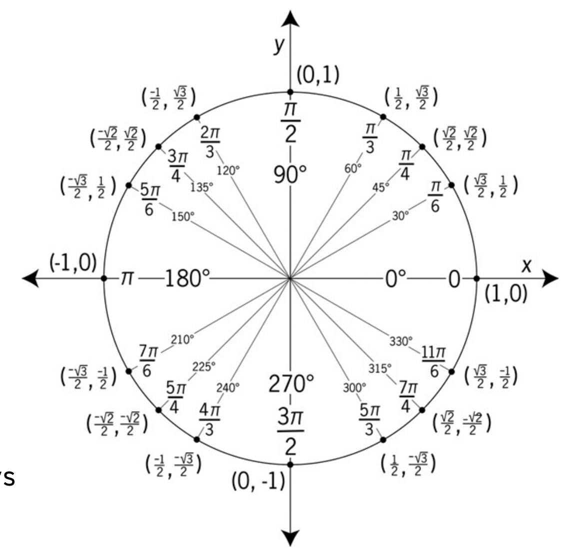


Starting Coordinate Estimation Heuristic

- Used training set timestamps to find last taxi ride in training set of each taxi ID.
- Assigned ending coordinates of last taxi ride (in training set) to starting coordinate of test set point via Nearest Neighbors.
- Distance between two points defined by timestamp difference.
- Calculated distance from city center as an extra feature.

Sine/Cosine for cyclical data

- We used a sine and cosine function to represent timestamp data as it is cyclic.
- Allows model to learn that 11pm and 1am are similar.
- Comes into play in test set with some data points on the border between two days (without adjusting for time zones).



EFFECTIVENESS OF NEAREST-NEIGHBOR HEURISTIC

RMSE	Estimated Location	No Estimated Location
Validation	549.311	555.17
Public Test	751.02716 Not bad!	762.08781

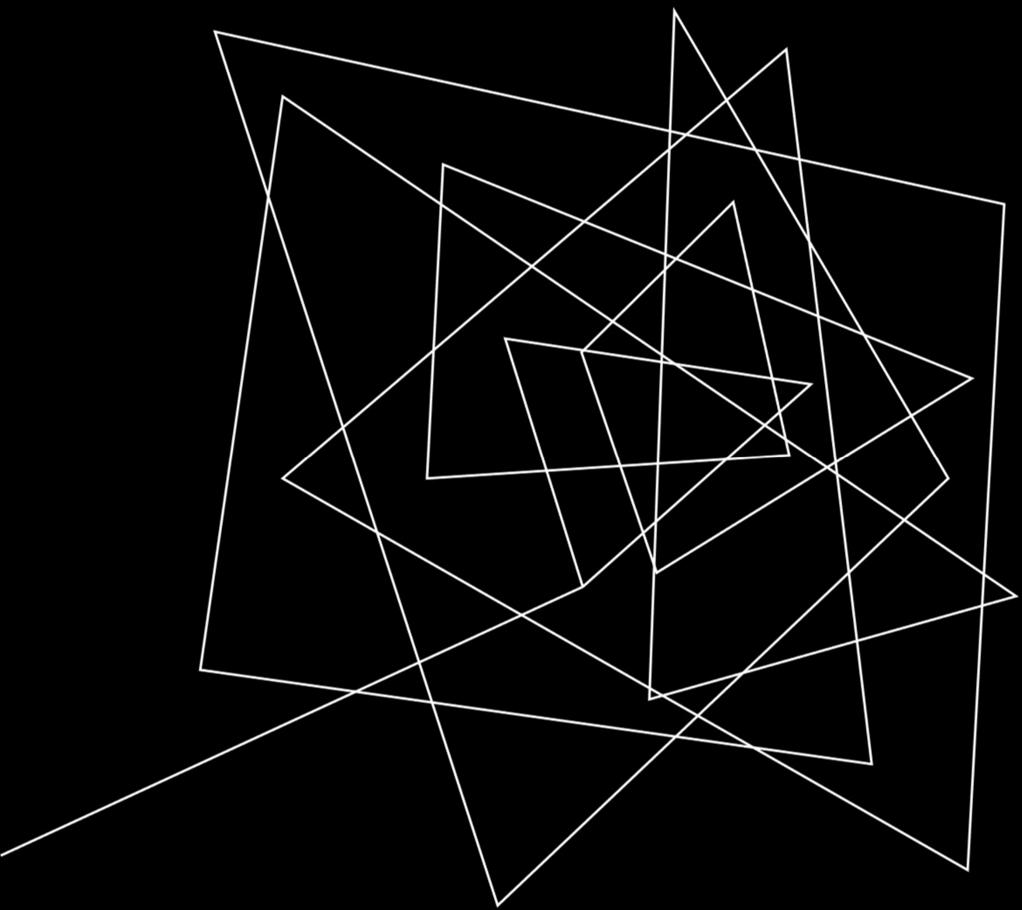
```
xgb_u = xgb.XGBRegressor(tree_method="gpu_hist",
                         booster="dart",
                         n_estimators=100,
                         enable_categorical=True,
                         max_cat_to_onehot=100,
                         objective="reg:gamma")
xgb_u.fit(train, train_label)
preds_xgbu = xgb_u.predict(df_test)
score(xgb_u)
```

Model Hyperparameters

FEATURES USED IN MODELS

Categorical	**Estimated Location**	**Time**	
ORIGIN_CALL	Start Longitude	Year	Day of week
ORIGIN_CALL	Start Latitude	Week of year	Hour of day
TAXI_ID	Start Distance	Month	+ Sin/cos of each time feature
Call Type		Day of month	

Categorical features were not used in sklearn models except for call type since it could be easily one-hot encoded.



METHODOLOGY

DEEP LEARNING MODELS

DEEP LEARNING MODEL (NO COORD)

Table 1: No-Coordinate Neural Network Architecture

Model Components		
Layer Name	Input	Output
Origin Call Embedding	Caller ID between 0 and 29026	20D vector embedding
Origin Stand Embedding	Origin Stand ID between 0 and 63	5D vector embedding
Taxi ID Embedding	Taxi ID between 0 and 447	10D vector
Linear Layer	$9 + 20 + 5 + 10 = 44$D vector	1000D vector
ReLU Layer	1000D vector	1000D vector
Dropout Layer	1000D vector (50% drop out)	1000D vector
Linear Layer	1000D vector	800D vector
ReLU Layer	800D vector	800D vector
Dropout Layer	800D vector (50% drop out)	800D vector
Final Linear Layer	800D vector	1D vector

776.30852	426.37***
Public Score (RMSE)	Validation (RMSE)

***Old version of validation set, $\mu + 5\sigma$ used as threshold, our non deep learning models outperform this model.

DEEP LEARNING MODEL (NO COORD)

TRAVEL_TIME
846.9913
708.6923
781.1971
676.8041
707.2174
830.4537
780.2921
808.3237
720.4258
766.7042
669.2481
793.9398
797.1658
1343.199
825.3787
863.8192
805.8036
948.8777
872.0022
911.3344
850.4429
819.1618
496.3416
850.4752
681.3383
784.0627

Its predictions are very monotonic

How can we push the model to make more aggressive predictions?

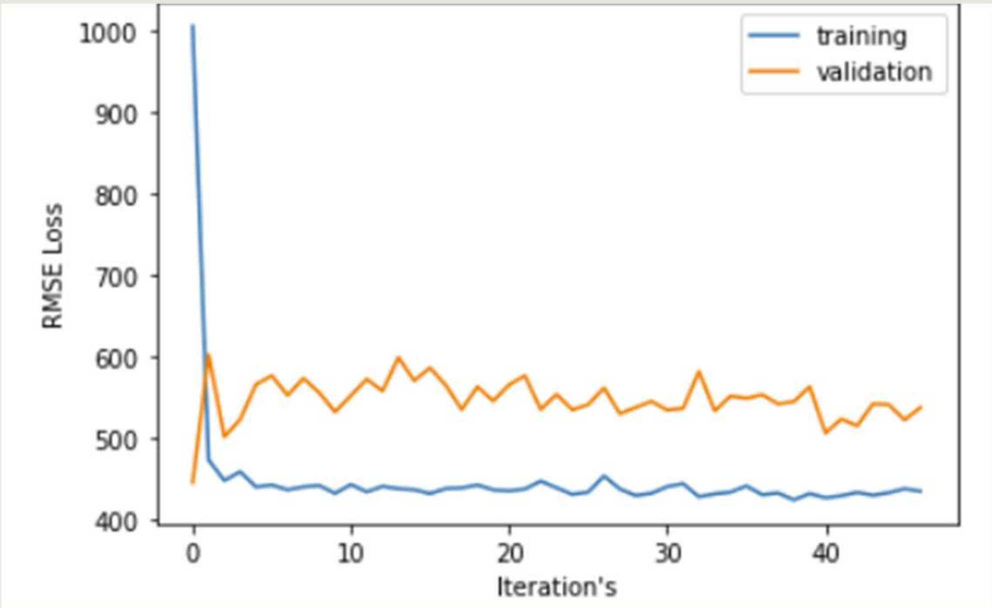
DEEP LEARNING MODEL (EMBEDDING)

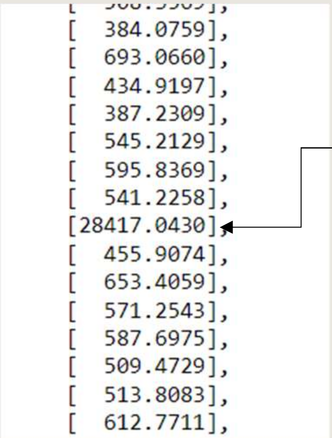
Table 2: Embedding neural network architecture

Model Components		
Layer Name	**Input**	**Output**
Origin Call Embedding	Caller ID between 0 and 29026	6D vector
Origin Stand Embedding	Origin Stand ID between 0 and 63	5D vector
Taxi ID Embedding	Taxi ID between 0 and 447	5D vector
Year Embedding	Year Between 0 and 1 (2013 or 2014)	2D vector
Week Embedding	Week of the year between 0 and 51	5D vector
Day embedding	Day of the week between 0 and 6	5D vector
Hour Embedding	Hour of the day between 0 and 23	5D vector
Linear Layer	$6 + 6 + 5 + 5 + 2 + 5 + 5 + 5 = 39$D vector	1000D vector
ReLU layer	1000D vector	1000D vector
Dropout layer	1000D vector (50% dropout)	1000D vector
Linear layer	1000D vector	1000D vector
ReLU layer	1000D vector	1000D vector
Dropout layer	1000D vector (50% dropout)	800D vector
ReLU layer	800D vector	800D vector
Linear layer	800D vector	1 number

TLDR: Everything is converted into embeddings, including time features

DEEP LEARNING MODEL (EMBEDDING)




```
[   384.0759],
[   693.0660],
[   434.9197],
[   387.2309],
[   545.2129],
[   595.8369],
[   541.2258],
[28417.0430],
[   455.9074],
[   653.4059],
[   571.2543],
[   587.6975],
[   509.4729],
[   513.8083],
[   612.7711],
```

WHAT IS THIS???

Interesting predictions on test set…

(This model sucks)

EXPERIMENTS

$$\frac{1}{n}\frac{\sum_{i=1}^{n} y^{(i)}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}{\sum_{i=1}^{n} y^{(i)}}$$

EXPERIMENT 1: WEIGHTED MSE LOSS FUNCTION (FOR NEURAL NETS)

- Goal: Get the model to make more aggressive predictions
- We used weighted mean square error as the loss function for neural nets
- Normally, the neural net just predicted the mean
- Now with the weights the predictions have more variance but are still clustered around the weighted mean

TRIP_ID	TRAVEL_TIME
T1	1171.141
T2	1022.262
T3	1072.797
T4	979.6191
T5	964.6397
T6	1184.334
T7	1047.993
T8	1253.434
T9	1026.705
T10	990.2237
T11	872.4241
T12	1170.919
T13	1199.347
T14	1982.23
T15	1210.193
T16	1325.64
T17	1113.413
T18	1451.198
T19	1156.466
T20	1326.213
T21	1099.164
T22	1155.411
T23	668.4753
T24	1117.39
T25	941.6244
T26	1019.368
T27	1077.721

EXPERIMENT 1: WEIGHTED MSE LOSS FUNCTION (FOR NEURAL NETS)

- Goal: Get the model to make more aggressive predictions
- We used weighted mean square error as the loss function for neural nets
- Normally, the neural net just predicted the mean
- Now the predictions have significantly more variance but are still clustered around the weighted mean

728.00536

735.16045

735.64185

719.322

729.36548

722.87803

EXPERIMENT 2: SIMPLE ENCODINGS

- We used one hot encodings for every single feature except starting and ending coordinates/distance
- We then fed the data into an ensemble of methods (Random Forests, KNN, Linear Reg, XGBoost, etc.)
- These generally performed the best on the public test set with fewer estimators. (4 was found to be best).
    - "Dumber" models performed better, makes us think there is a major distribution shift on the test set
- TLDR: XGBoost and "dumber" models performed best in public score.
- We don't trust these models and therefore we will mostly ignore public RMSE for our final submissions

EXPERIMENT 3: OTHER MODELS

RMSE	Linear Regression	Gradient Boosting	Random Forest	XGBoost	XGBoost (Stacked)	Best Simple Model (Prev. slide)	Blind predicting the mean :/
Validation	581.72	565.47	575.98	549	539.95	586.78	599.412
Public Test	N/A	N/A	N/A	751.03	796.64	719.322	786.72283

XGBoost significantly outperformed all other model classes in validation RMSE and public test RMSE. Its predictions were also less monotonic.

And many more approaches that were too much to fit in a presentation...

EXPERIMENT 4: USE SNAPSHOT INFORMATION

YEAR	WK_OF_YR	WK_DAY	MONTH	DAY	HR
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10
2014	33	3	8	14	10

- All points in the test set came from certain snapshots.
- We trained several models on similar times of day/times of year using the same XGBoost hyperparameters from slide 11.
- If there weren’t many similar samples in the training set, we opted to use our baseline model (Best XGBoost model on whole dataset).

OUR ACTUAL PREDICTIONS

- We decided to roll with our standard XGBoost model. It had the best combination of validation RMSE (549) and public test RMSE (751).
- Our second submission is our snapshot ensemble predictions, as we thought specialized models might be able to create more aggressive predictions.

```
xgb_u = xgb.XGBRegressor(tree_method="gpu_hist",
                         booster="dart",
                         n_estimators=100,
                         enable_categorical=True,
                         max_cat_to_onehot=100,
                         objective="reg:gamma")
xgb_u.fit(train, train_label)
preds_xgbu = xgb_u.predict(df_test)
score(xgb_u)
```

```
# Christmas one
spec_df.append(df_train[(df_train["MONTH"] == 12) & (df_train["DAY"] > 18)])
xgb_x = xgb.XGBRegressor(tree_method="gpu_hist",
                         booster="dart",
                         n_estimators=100,
                         enable_categorical=True,
                         max_cat_to_onehot=100,
                         objective="reg:gamma")
xgb_x.fit(spec_df[0][['ORIGIN_CALL','ORIGIN_STAND', 'TAXI_ID', 'START_LONG', 'START_LAT',
       'A', 'B', 'C', 'YEAR', 'WK_OF_YR', 'WK_DAY', 'MONTH', 'DAY', 'HR',
       'DIST', 'WK_OF_YR_SIN', 'WK_OF_YR_COS', 'WK_DAY_SIN', 'WK_DAY_COS',
       'MONTH_SIN', 'MONTH_COS', 'DAY_SIN', 'DAY_COS', 'HR_SIN', 'HR_COS']], spec_df[0]["TARGET"])
preds_xgbx = xgb_x.predict(df_test[df_test["MONTH"] == 12])
```

DISCUSSION

WHAT WE LEARNED

Neural Nets Suck (At tabular data)

- After lots of experimentation with neural nets and PyTorch, we realized neural nets suck for tabular data.
- Most of the neural nets were just glorified mean predictors that took too long to train.

Generalization in Machine Learning

- There seems to be some covariate shift between training and public test data.
- Training data has very little predictive power in this competition compared to ML applied to other tasks.
- Lots of useless columns/features (missing data, day type, trip id).

XGBoost Good

- Out of all the models, XGBoost worked best for both validation and public test set results.
- Also trained a lot faster than other methods (gpu acceleration)
- We are now worshipping XGBoost as our new God.

FUTURE WORK

- We could experiment more with the sequential nature of taxi rides.
- Take into account how taxis tend to go back towards the city center after a taxi trip for better starting coordinate estimation.
- Try more sophisticated methods of blending predictions from multiple models.

THANK YOU

ENJOY SOME RANDOM CHICKENS
:D