Using Machine Learning to Predict New York City Agency Response Times

Mariia Aleksandrovych and Rebecca Heilweil

City agencies vary sharpy in response times

911 calls are (ideally) answered relatively quickly. But 311 calls, which might be used to address everything from illegal parking and heat issues to abandoned cars, don’t necessarily have the same sort of reliability.

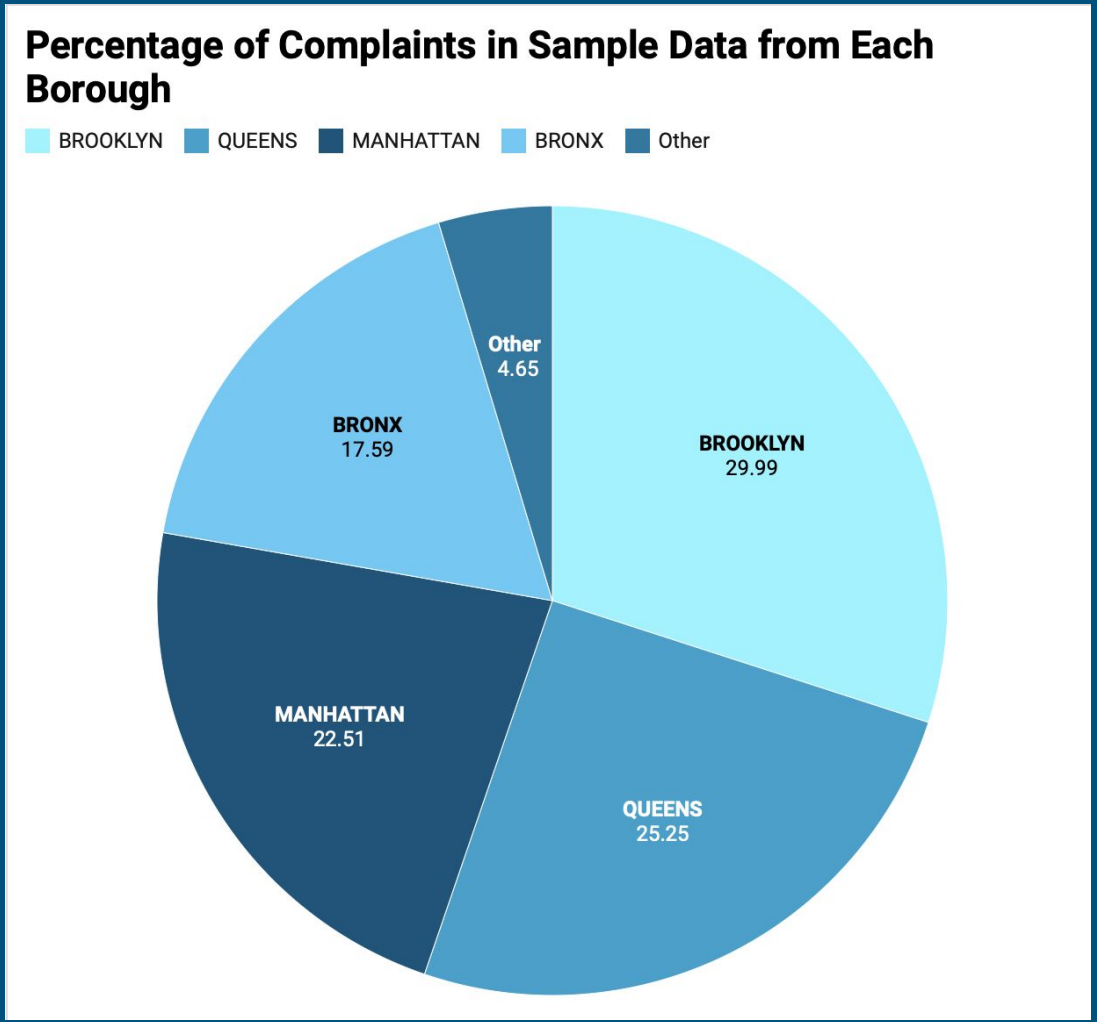
Our project aims to evaluate which machine learning based systems might be best suited toward predicting how long city agencies take to respond to calls from constituents. **Ultimately, we create a model that predicts how fast your issue might get resolved by the city. Our system outperforms linear regression.**

Why does this matter

- We are interested in providing a view into the efficacy of New York City agencies. Understanding how fast an agency might respond to an issue can help inform people, in real time, about how reliable the city might be.
- This research might also give us insight into whether the city is serving some locations, or certain types of complaints, more quickly than others.
- We might find errors in the data to inform city officials — we actually did find potential errors!

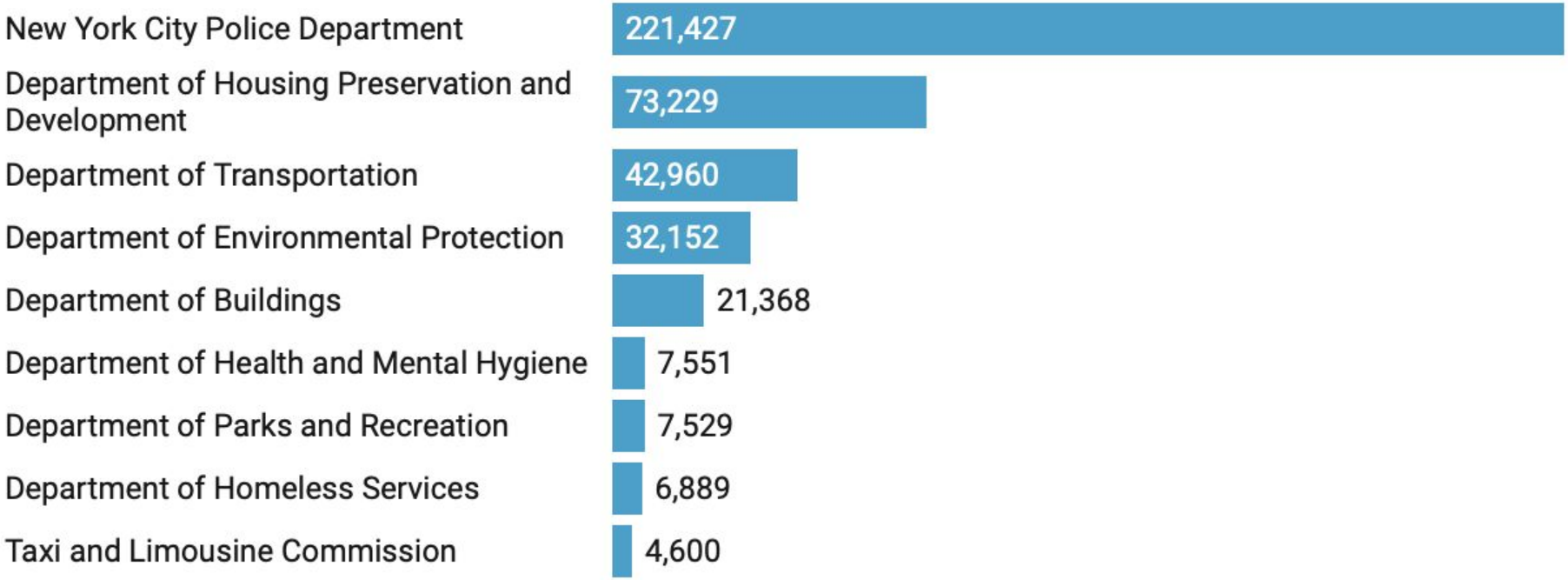
Our data: New York's Open Data repository

- As part of the NYC Open Data initiative, New York City has been collecting data on 311 response times since 2010.
- Updated daily, this database includes more than 35 million requests that have been called into 311 — ensuring there is plenty of data available for testing, training, and validation.
- Features include location, the type of complaint, the responding agency, a description of the complaint, borough.
- This data is produced automatically.
- We worked with about 500,000 samples.


Percentage of Complaints in Sample Data from Each Borough
BROOKLYN
QUEENS
MANHATTAN
BRONX
Other
Other
4.65
BRONX
17.59
BROOKLYN
29.99
MANHATTAN
22.51
QUEENS
25.25

Agencies receiving the greater number of 311 reports

For visualization purposes, we only included agencies with greater than 4,000 complaints, though the others are included in the system.



Issues with the Longest Response Times

This is based on the City Council's' representation of this data, which is possibly larger than what we were able to train our models on.

Feature selection

There were about 40 options for features. Many of these features were redundant (for instance, there were multiple cells for longitude and latitude, as well as two different columns representing the city agency responding).

Some of these features were almost always null because they only applied to very specific circumstances (like 311 requests that involved a taxi service).

We also remove a call ID number, since that number was essentially meaningless.

We converted two dates, the date a complaint was created and the date it was resolved, into a new metric: hours passed between complaint created and resolved.

Error in NYC Open Data

While completing our research, we discovered that at least 2,000 — we are working with sample data, so the real number may be higher — reports filed by or about the Department of Transportation have errors.

We discovered this issue because we created a “Time Elapsed” feature from a “Created” and “Resolve” data feature — and realize we were getting theoretically impossible negative time elapsed values.

We decided to drop any scenario in which this happened, but future research could involve alerting NYC DOT of the issue and investigating.

Baseline models, based on MSE

We evaluate three primary baselines:

Baseline Mean Squared Error, calculated directly from the dataset. **~32006**

MSE using a Lasso Regression: **~29034**

MSE using Linear Regression **~29028**

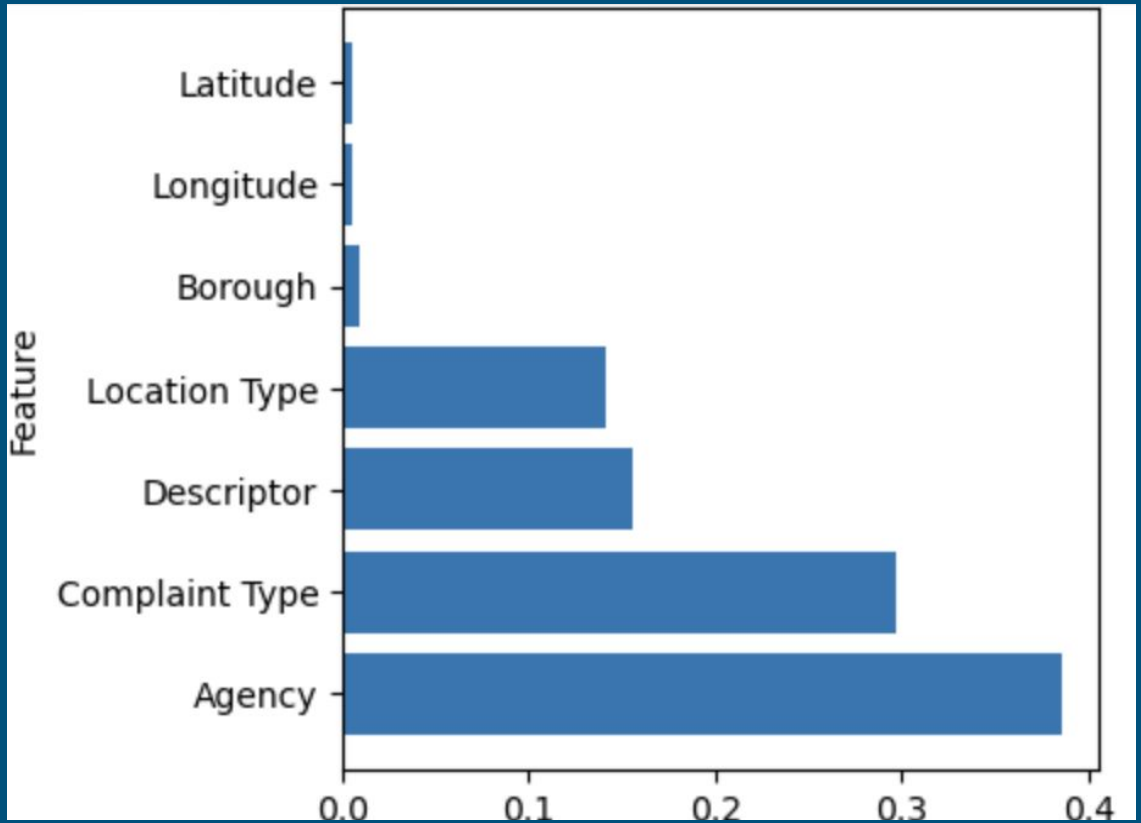
First Model: Gradient Boosted Tree Regression

Using default parameters, we achieve an **MS**E of ~16000, noticeably **outperforming out baselines.**

We also saw that agency was highlight related to the response time, which we measured in hours, followed by complaint type, descriptor, and then location type.

Interestingly, Borough, Longitude, and Latitude did not provide significant importance to the model

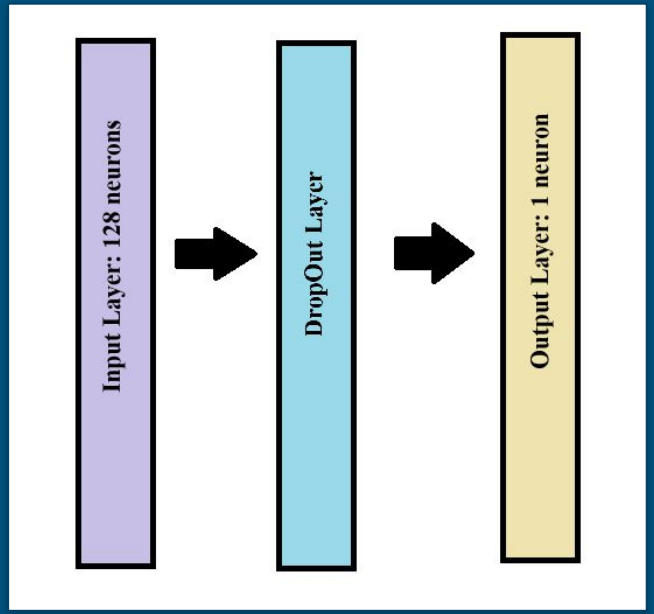
First Model: Features Importance



Hyperparameter Tuning for GBT Regressor

	N_Estimators	Learning Rate	Depth	R2
0	300	0.30	7	0.732853
0	300	0.10	7	0.711626
0	300	0.30	5	0.705243
0	100	0.30	7	0.703394
0	50	0.30	7	0.679129
0	100	0.10	7	0.667795
0	300	0.10	5	0.658996
0	100	0.30	5	0.657224
0	50	0.10	7	0.642289

Second Model: Neural Network



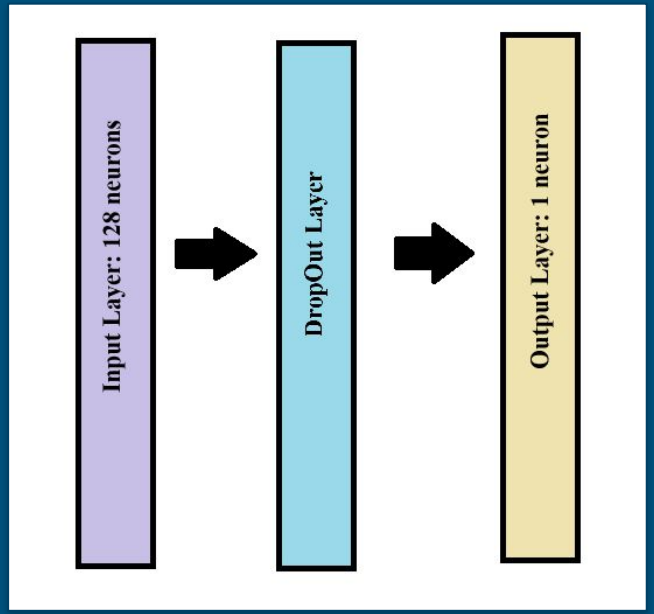
Architecture.

1. **Input layer:** fully connected 128 neurons, ReLU activation, L2 regularization, and input shape (, 7) i.e. seven features.
2. **DropOut Layer:** Drops 20% of the passed data.
3. **Output Layer:** Single neuron, Linear activation

Training

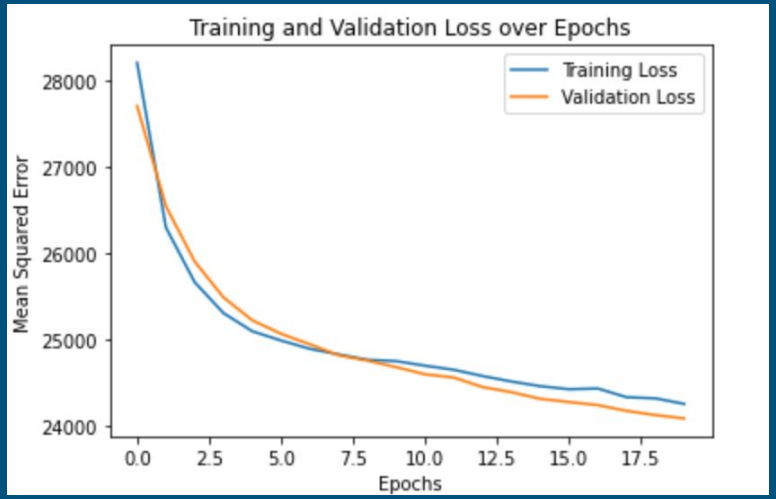
- **Optimizer:** Adam
- **Learning Rate:** 0.001 for 10 epochs after 10 epochs we have exponential decay: Learning Rate(epoch)=0.001×exp(0.1×(10−epoch))
- **Batch size:** 32
- **Validation split**: 80% training, 20% validation
- **Loss Function:** Mean Squared Error
- **#N: ~11k**

Second Model: Neural Network



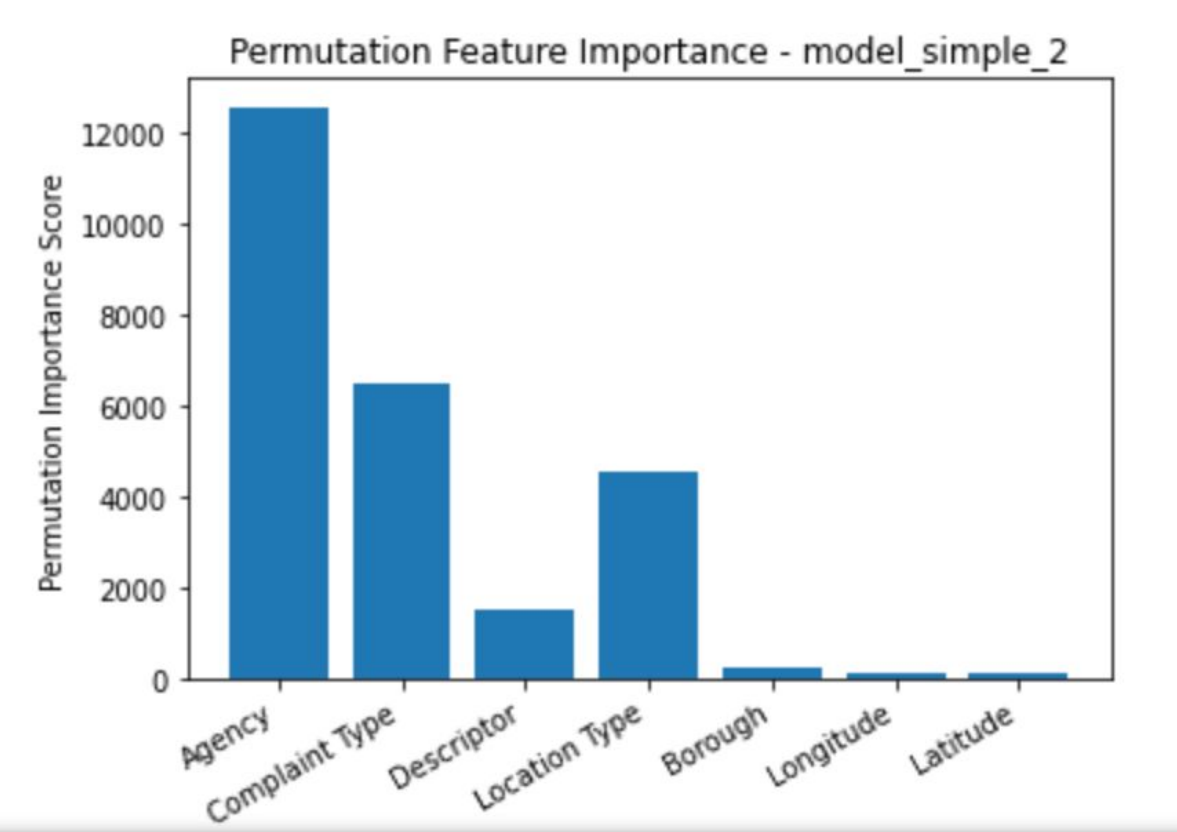
Performance:

- Mean Squared Error on the training data set: 23,253
- Mean Squared Error 5 fold cross-validation: 20,861 +/- 600
- Mean Squared Error on out of sample set (~10,000): 65,427

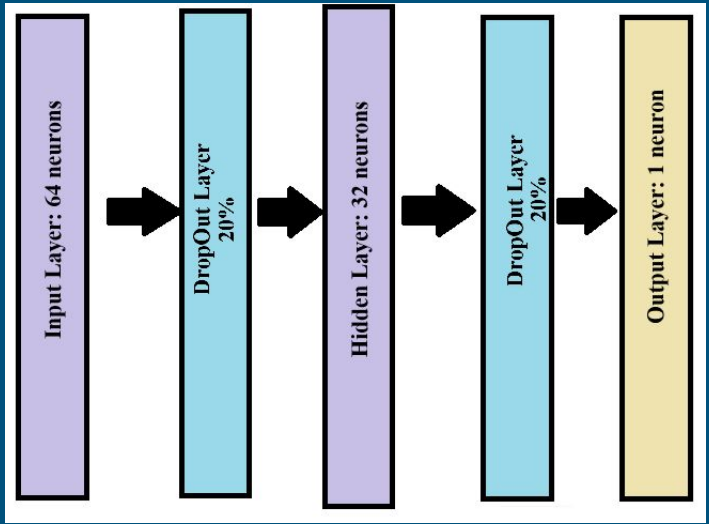


- Baseline (mean): 71,167 Baseline (Linear regression): 66,329

Second Model: Features Importance (Permutations)



Third Model: 2 Layers Neural Network



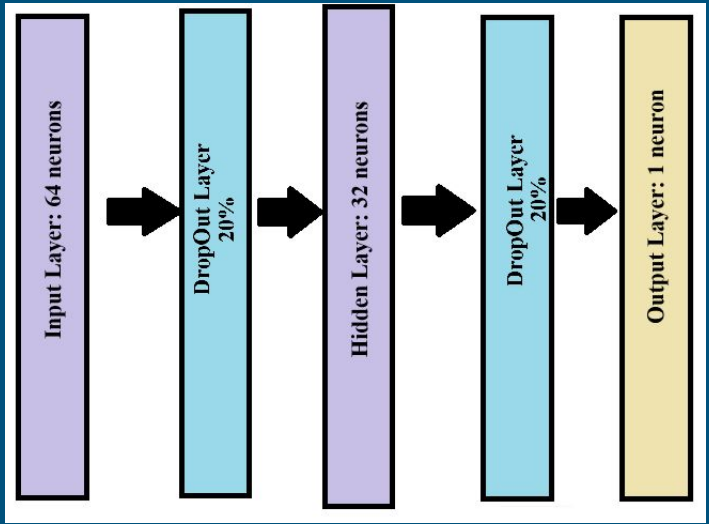
Architecture.

1. **Input layer:** fully connected 64 neurons, ReLU activation, L2 regularization, and input shape (, 7) i.e. seven features.
2. **DropOut Layer:** Drops 20% of the passed data.
3. **Hidden Layer:** fully connected 32 neurons, ReLU activation, L2 regularization, and input shape (, 7) i.e. seven features.
4. **DropOut Layer:** Drops 20% of the passed data.
5. **Output Layer:** Single neuron, Linear activation

Training

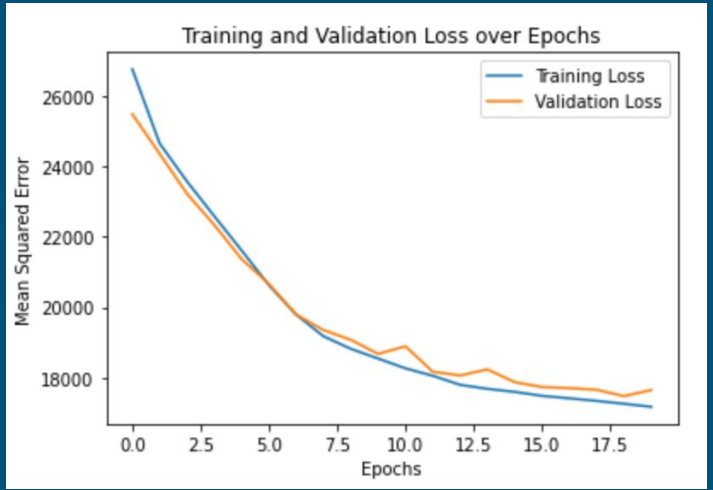
- **Optimizer:** Adam
- **Learning Rate:** 0.001 for 10 epochs after 10 epochs we have exponential decay: Learning Rate(epoch)=0.001×exp(0.1×(10−epoch))
- **Batch size:** 32
- **Validation split**: 80% training, 20% validation
- **Loss Function:** Mean Squared Error
- #N: ~29k

Third Model: 2 Layers Neural Network



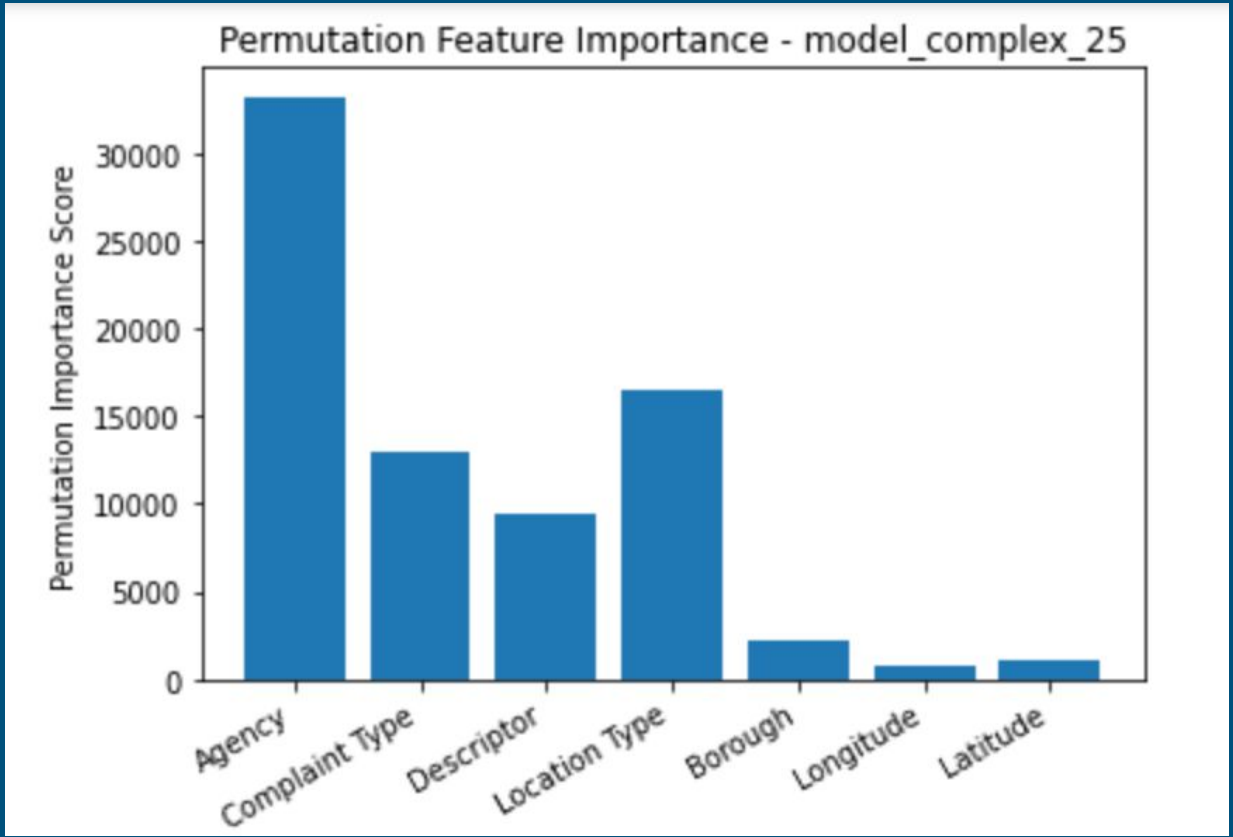
Performance:

- Mean Squared Error on the training data set: 16,870
- Mean Squared Error 5 fold cross-validation: 16,587 +/- 873
- Mean Squared Error on out of sample set (~10,000): 67,353

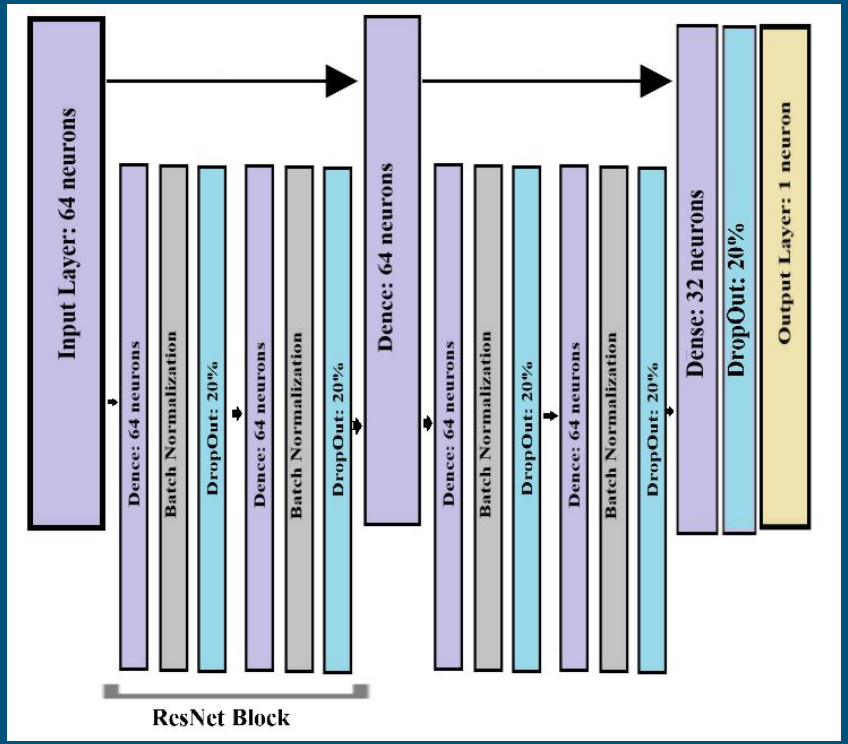


- Baseline (mean): 71,167 Baseline (Linear regression): 66,329

Third Model: Features Importance (Permutations)



Fourth Model: Deep ResNet



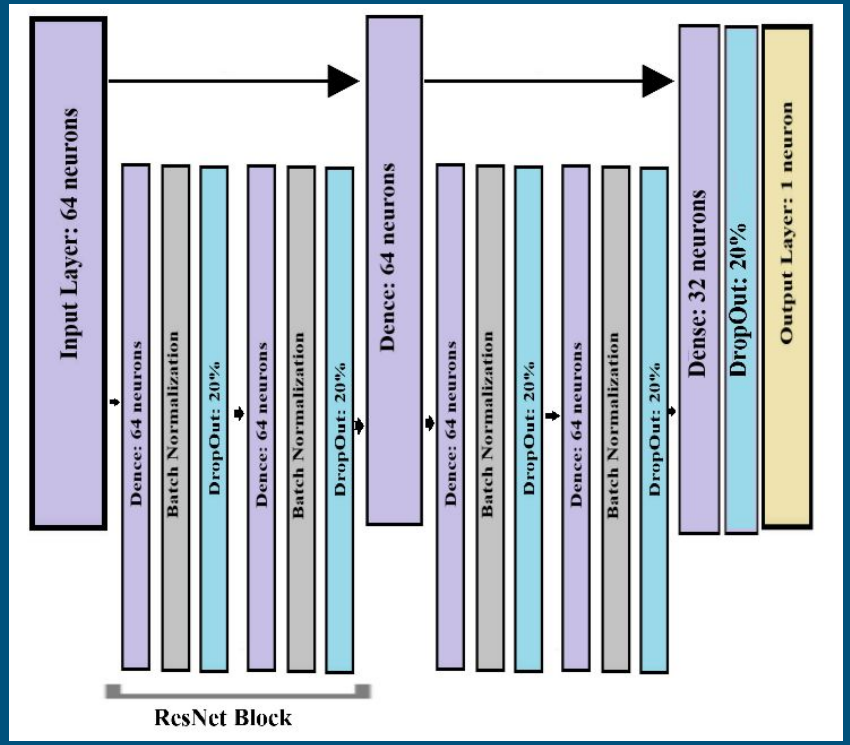
Architecture.

1. **Input layer:** fully connected 64 neurons, ReLU activation, L2 regularization, and input shape (, 7) i.e. seven features.
2. **Residual Block: 2** Dense layer (64 neurons, ReLU activation, L2 regularization) following with Batch Normalization, and Dropout (20%) Layers
3. **Dense:** fully connected 64 neurons, ReLU activation, L2 regularization
4. **Second Residual Block: 2** Dense layer (64 neurons, ReLU activation, L2 regularization) following with Batch Normalization, and Dropout (20%) Layers.
5. **Dense:** fully connected 32 neurons, ReLU activation, L2 regularization
6. **DropOut Layer:** Drops 20% of the passed data.
7. **Output Layer:** Single neuron, Linear activation

Training

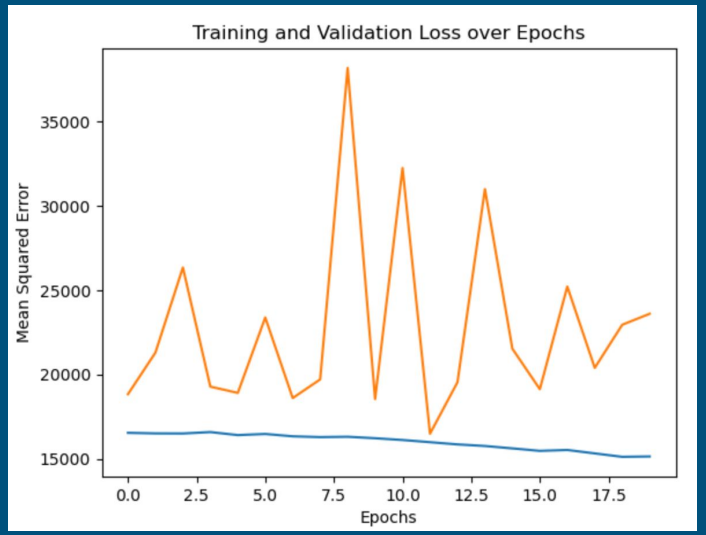
- **Optimizer:** Adam
- **Learning Rate:** 0.001 for 10 epochs after 10 epochs we have exponential decay: Learning Rate(epoch)=0.001×exp(0.1×(10−epoch))
- **Batch size:** 32
- **Validation split**: 80% training, 20% validation
- **Loss Function:** Mean Squared Error
- ~#N: ~47k

Fourth Model: Deep ResNet



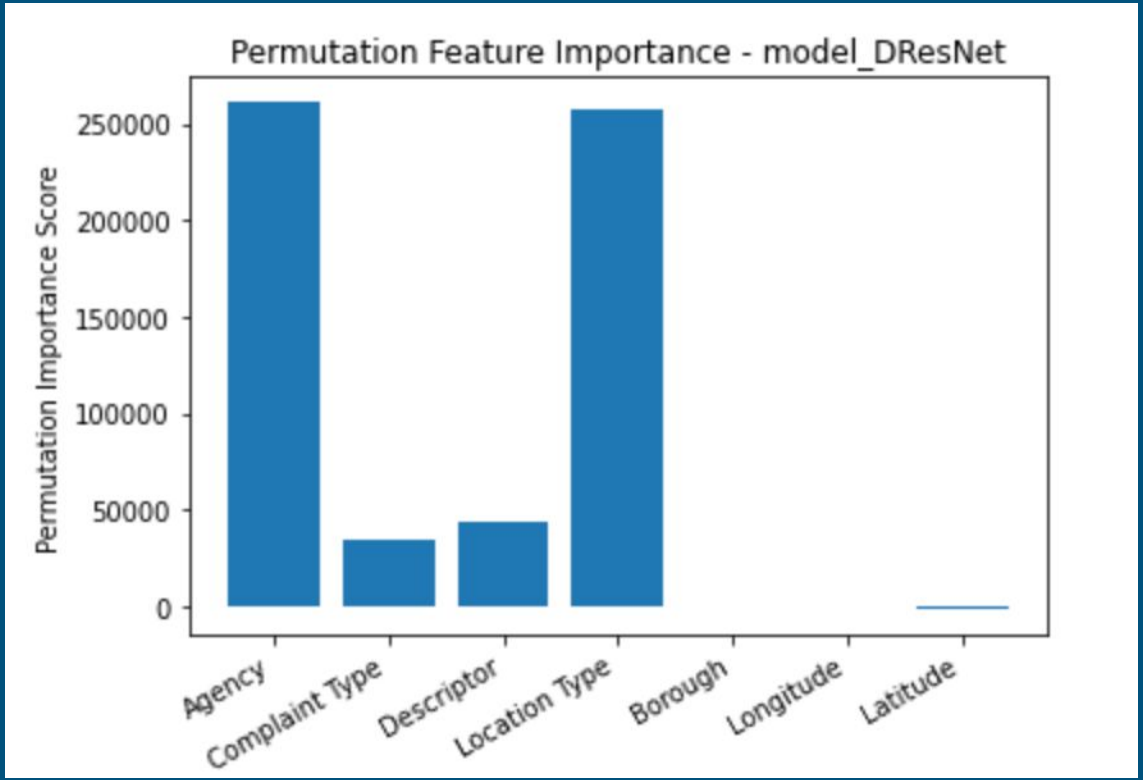
Performance:

- Mean Squared Error on the training data set: 17,900
- Mean Squared Error 5 fold cross-validation: 19,422 +/- 3,547
- Mean Squared Error on out of sample set (~10,000): 84,365



- Baseline (mean): 71,167 Baseline (Linear regression): 66,329

Fourth Model: Deep ResNet



5 fold Cross-Validation (MSE)

Linear Regression - MSE: 28904.17261398608 +/- 346.5560153708919
Lasso Regression - MSE: 28918.12591391247 +/- 343.8195874035514
GB Regression - MSE: 4985.333993772696 +/- 86.11027934727582
Simple 1 layers NN- MSE: 20861.137890625 +/- 600.5459880750432
2 layers NN - MSE: 16587.6294921875 +/- 381.237108979759
3 layers DResNet - MSE: 59389.391796875 +/- 84009.95419371294

Out-of-sample comparative performance (MSE)

MSE for Baseline Model: 71167.0791123987
MSE for Linear Regression Model: 66329.90822792749
MSE for Lasso Model: 66449.24024717495
MSE for GBT Model: 64077.68799608705
MSE for NN 1 layers Model: 63790.3143967668
MSE for NN 2 layers Model: 65652.52003491706
MSE for DResNet 3 layers Model: 84365.82704451475

*We ran this several times and got different results, but the GBT and 1-layer NN always tended to outperform the others. The file size is too big to effectively shuffle.

Reflections:

It makes sense that this system performed far better on tabular data — we know that neural networks don't necessarily perform well.

It's interesting that agency has such a high impact on 311 call response rate, but, perhaps more importantly, **location is not a primary driver of response times using most models.**

The model is probably useful for determining if 311 will fix your issue in a few hours, half a day, or several days — but can't necessarily give you an hour-by-hour prediction.

It's interesting we got one neural network to perform basically as well as a Gradient Boosted Tree regressor!