Used Car Price Prediction

By Ricky, Sameer, Sam

Introduction & Relevance

Goal: Predict prices of used cars using various features from a Kaggle dataset.

Relevance:

- Financial planning for sellers and buyers.
- Platforms like Kelly Blue Book provide similar insights.
- Sellers capture maximum value; buyers get reasonable prices.
- Understand depreciation trends.
- Insurance companies set premiums and manage risk.
- Banks issue appropriate loan amounts and prevent defaults.

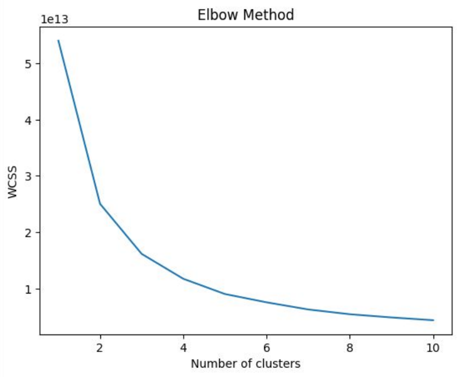
Data

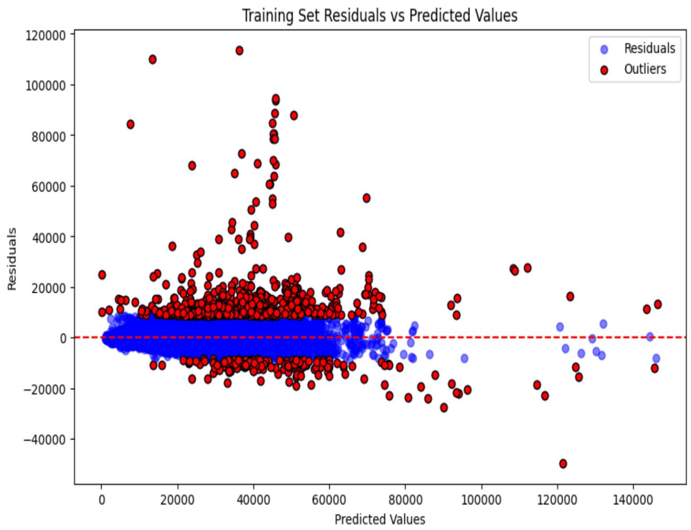
- Kaggle dataset that was originally obtained from scraping used car listings: https://www.kaggle.com/datasets/adityadesai13/used-car-dataset-ford-and-mercedes/data?select=vw.csv
- The data is split up by brand, concatenated into one dataframe totaling ~100,000 rows
- **Features**: Brand, price, engine size, tax, mileage, transmission, fuel type, mpg, model, year

Data

Encoding

- Label encoding
- Frequency / Ordinal Encoding
- One-Hot Encoding



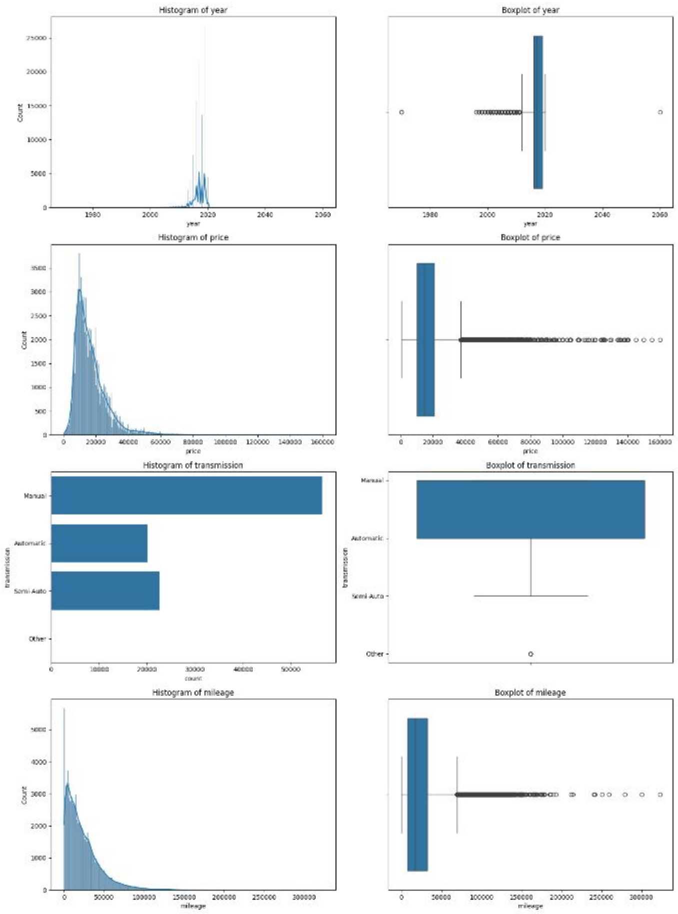


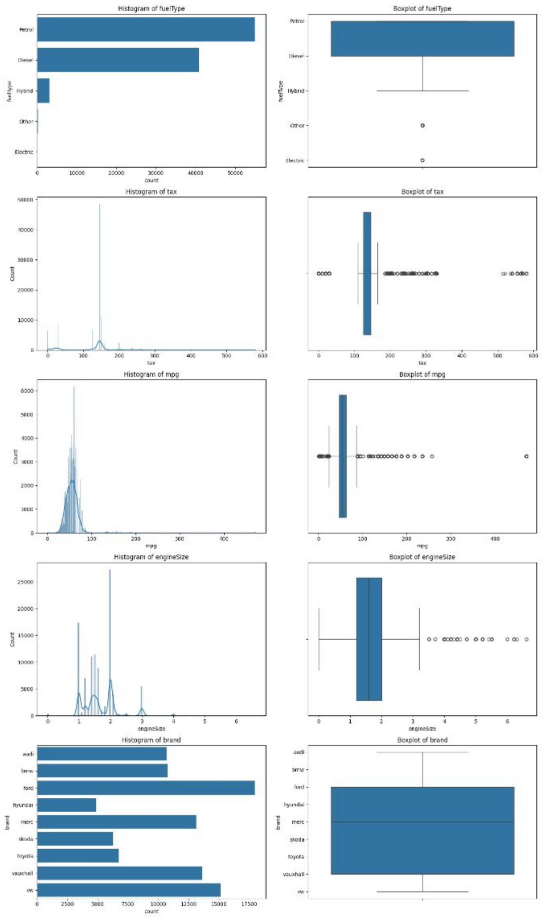
KMeans Cluster Average Imputation

- Elbow Method - plot WCSS for each number of centroids tested
- Impute 0’s in columns with cluster averages

Removing Outliers

- Get rid of data points that are more than 3 standard deviations away from the mean difference in training prediction and label


Histogram of year
Boxplot of year
Count
year
Histogram of price
Boxplot of price
price
Histogram of transmission
Boxplot of transmission
Manual
Automatic
Semi-Auto
Other
transmission
count
Histogram of mileage
Boxplot of mileage
mileage

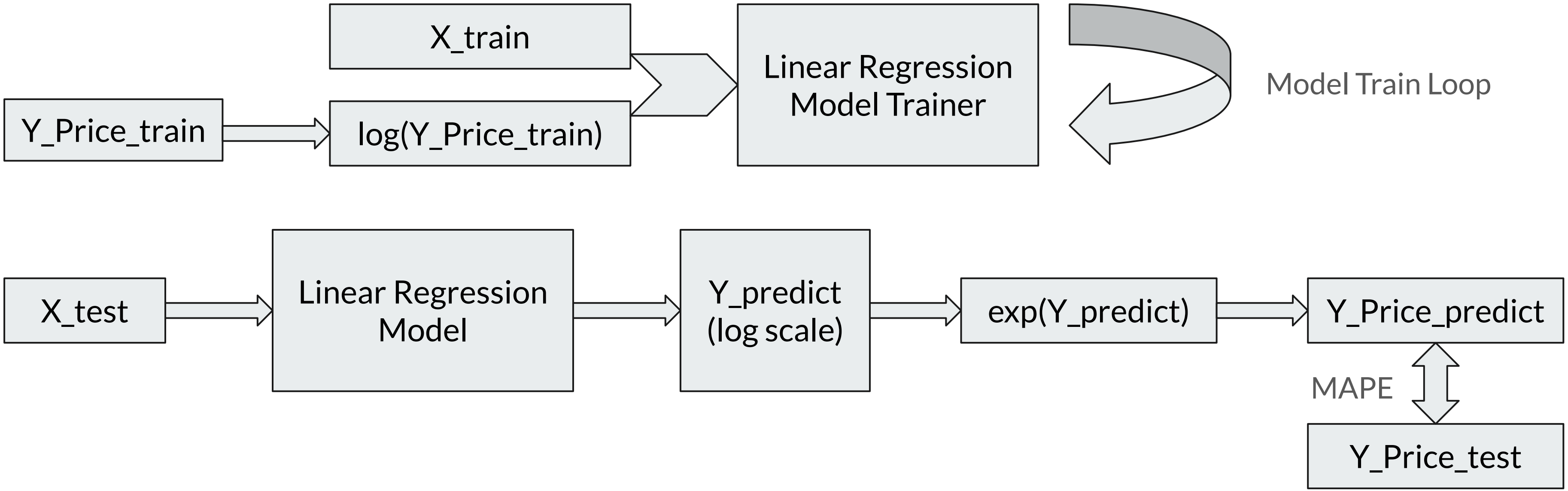

Histogram of fuelType
Boxplot of fuelType
Petrol
Diesel
Hybrid
Other
Electric
fuelType
count
Histogram of tax
Boxplot of tax
tax
Histogram of mpg
Boxplot of mpg
mpg
Histogram of engineSize
Boxplot of engineSize
engineSize
Histogram of brand
Boxplot of brand
audi
bmw
ford
hyundai
merc
skoda
toyota
vauxhall
vw
brand

Modeling & Metric Choice

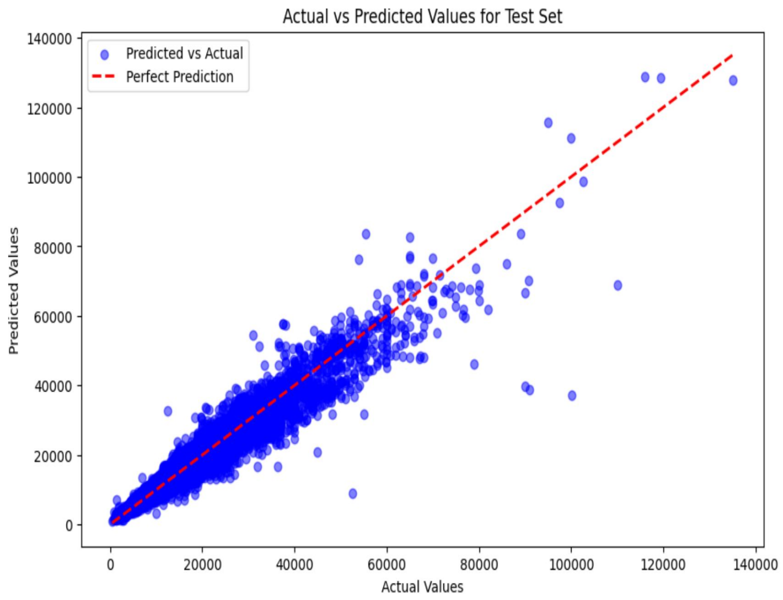
- Linear Regression
  - Baseline (numerical features only) resulted in 28% MAPE value
  - Simple and interpretable
  - Quick prototyping
  - Low variance
- Random Forest Regressor
  - Robust and stable
  - Non-linear
  - Various data types
- XGBoost
  - Efficient
  - Regularization built in

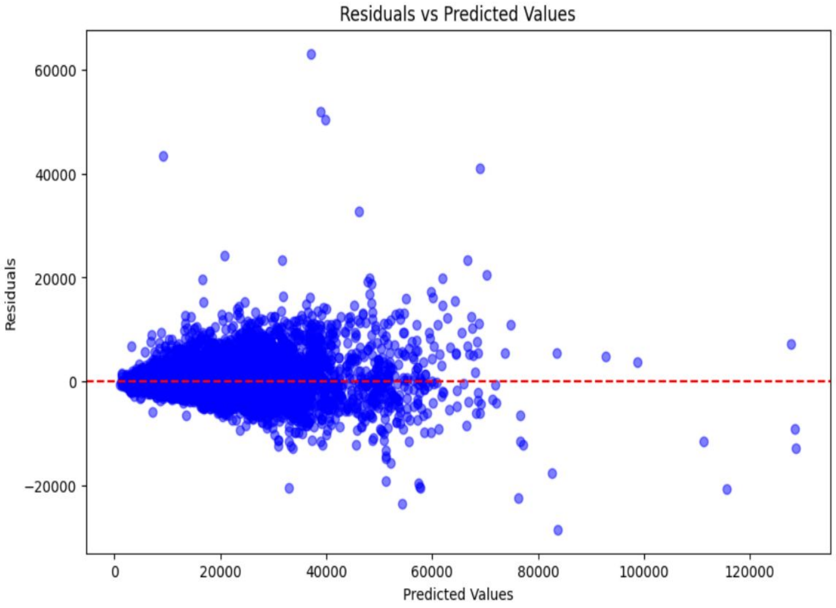
- MAPE
  - Strong interpretability
  - Good for comparing performance across models
- RMSE (Root Mean Squared Error): Penalizes larger errors more than MAE, useful for emphasizing the impact of significant prediction errors.
- R^2 (R-squared): Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables, showing model fit quality.

Linear Regression - Log transform

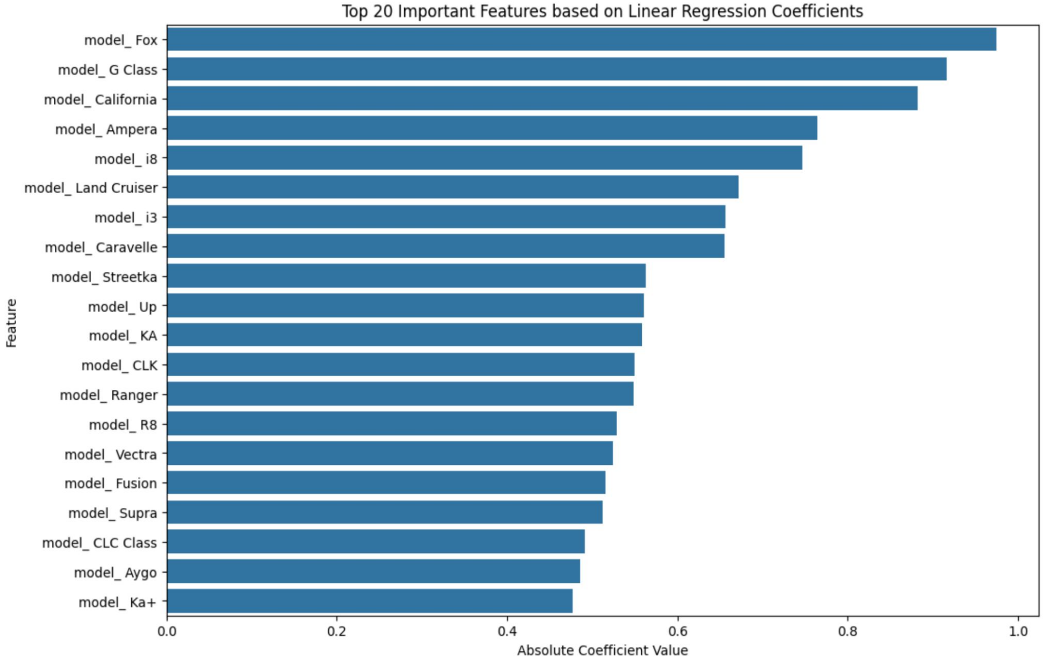


Linear Regression - Plot





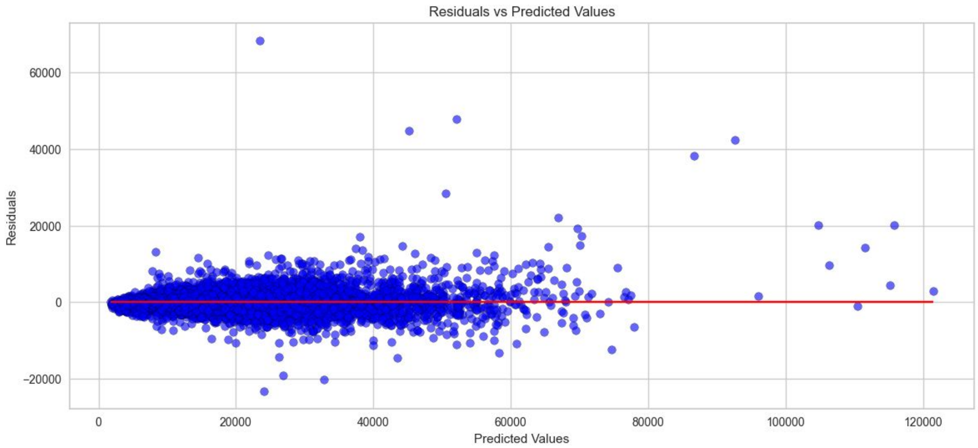
Linear Regression - Parameters

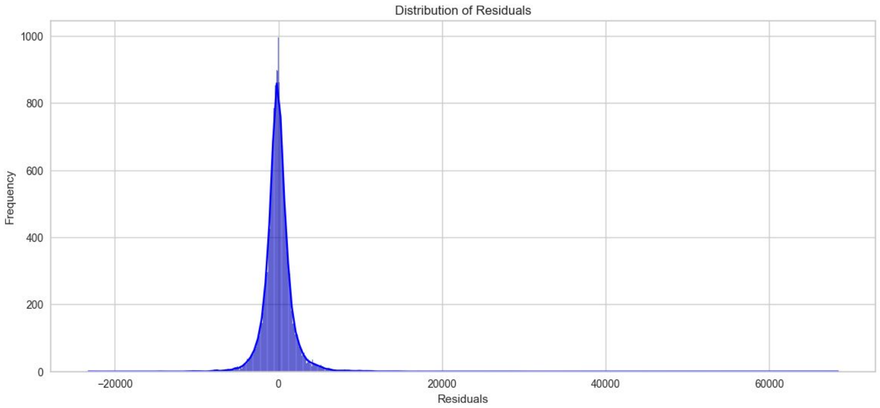


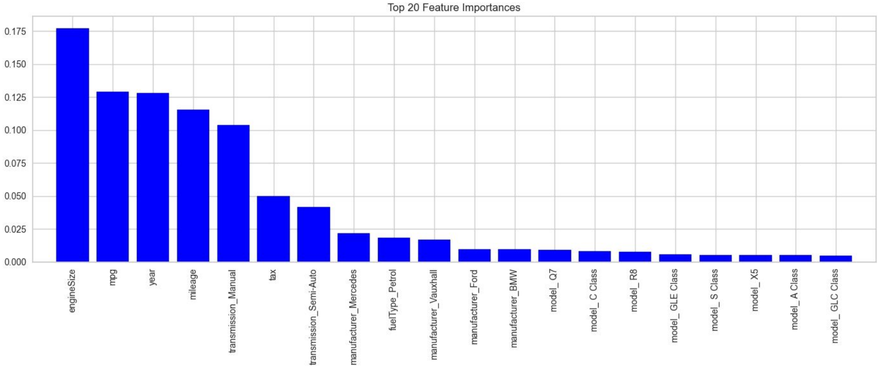
Intercept: 9.96

MAPE	MSE	R2
9.64%	6759186	0.93

Random Forest Regressor







Random Forest Regressor - Results & Further Steps

Model	MAPE	Best Parameters
RSCV	7.39%	`'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 50, 'bootstrap': False`
GSCV	7.30%	`'bootstrap': False, 'max_depth': 50, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 8, 'n_estimators': 150`
Cross-Validation	7.50%	NA

XGBoost Performance & Cross-Validation

	MAPE
Without K-fold Cross-Validation	6.55%
With K-fold Cross-Validation (k=5)	6.67%

- What are folds? - the data is split into a certain number of equal parts, called folds. For example, in 5-fold cross-validation, the data is split into 5 folds.
- The model is trained on some of these folds and tested on the remaining fold. This process is repeated so that each fold is used as the test set exactly once.
- Using folds ensures that every data point is used for both training and testing, which gives a more accurate estimate of the model's performance on unseen data. It helps prevent overfitting and ensures the model generalizes well.

Experiments - XGBoost Tuning

Learning Rate (eta)

Number of Boosting Rounds (n_estimators)

Max Depth (max_depth)

Min Child Weight (min_child_weight)

Subsample

Colsample_bytree

Gamma (min_split_loss)

```
# Set up the parameter grid
param_grid = {
    'learning_rate': [0.01, 0.1],
    'max_depth': [15, 25],
    'min_child_weight': [1, 5],
    'subsample': [0.6, 1.0],
    'colsample_bytree': [0.6, 1.0],
    'n_estimators': [100, 300]
}
```

XGBoost - Explanation

- XGBoost creates (n_estimators) decision trees that reach a max depth by splitting to decrease loss
- Each Tree evaluates potential splits based on features (determined by col_sample_by_tree, subsample) in the dataset and continues until max depth is reached - or until no further loss reduction can be achieved
- The child nodes from the split must contain a number of datapoints greater than min child weight
- Take mileage for example:
    - The model tests a split on mileage > 50,000
        - Take the average of all the prices > 50,000
        - Take the average of all the prices <= 50,000
            - Once it has the average price of all the cars with less than 50,000 miles it then calculates how far off it would be if it predicted every car with less than 50,000 miles having that price vs the actual price
            - Repeat for >50,000 miles
            - If the predictions reduce loss (by a value greater than gamma) the split is made
    - The model may choose to go through with the split when the loss is reduced (gamma is positive)

Modeling - Comparisons (Below from Kaggle)

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Random Forest	1243.373600	4770456.413400	2174.404000	0.954700	0.106400	0.073000
1	Extra Trees Regressor	1268.165800	4847329.500100	2192.978500	0.953900	0.108300	0.074700
2	K Neighbors Regressor	1319.478400	5381127.718600	2312.093100	0.948900	0.113400	0.077800
3	CatBoost Regressor	1533.765800	6398008.769500	2522.147900	0.939100	0.115700	0.086400
4	Light Gradient Boosting Machine	1525.920700	6811076.101400	2601.437900	0.935300	0.116900	0.086200
5	Decision Tree	1537.566200	7193454.251300	2675.451300	0.931500	0.136000	0.091600
6	Bayesian Ridge	1979.942500	11510906.833700	3378.601300	0.890600	0.161400	0.117600
7	Ridge Regression	1980.975100	11526146.446200	3380.944000	0.890500	0.161600	0.117700
8	Extreme Gradient Boosting	2123.320800	11899982.018300	3444.910900	0.886600	0.155700	0.119500
9	Gradient Boosting Regressor	2125.838400	11934059.716700	3449.592800	0.886300	0.155600	0.119600
10	Orthogonal Matching Pursuit	2514.289500	18350749.090700	4277.231900	0.825300	0.200100	0.149300
11	Passive Aggressive Regressor	2634.800500	19332661.577900	4325.778700	0.817600	0.205200	0.153300
12	AdaBoost Regressor	3891.793600	30608766.371200	5528.577600	0.707700	0.283900	0.254300
13	Elastic Net	6637.053500	95509973.220900	9768.712200	0.088600	0.493300	0.443000
14	Lasso Regression	7235.917300	109457878.190600	10458.493600	-0.044700	0.540400	0.494100

Performance Evaluation

MAPE	Baseline	Linear Regression	XGBoost	Random Forest
Without K-fold Cross-Validation	27.78%	9.64%	6.55%	7.24%
With K-fold Cross-Validation (k=5)		9.63%	6.66%	7.42%

Conclusions

- We chose a good dataset
- Accuracy can be improved by filter out the outlier of the dataset using statistic knowledge
- We cleaned and processed the data
- We encoded and scaled the data
- We experimented with a bunch of models and did some feature engineering
- Once we found a model of choice we tuned the model until we exceeded Kaggle standards
- We built a predict.py script using the encoder, scaler, and model artifact

Code Submission & Contributions

- https://github.com/sameer-karim/w207-final-project/tree/project
- Dataset - Sameer, Sam
- EDA - Sameer, Sam, Ricky
- Linear Regression, Feature Engineering - Sam
- Other Models - Sameer, Sam, Ricky
- Cross-Validation - Sam, Sameer
- ML Pipeline - Sam
- Presentation, Repo, and ReadMe - Ricky, Sameer, Sam

Final presentation. Your slides should include: (12 minutes + 2 minutes total (no Q&A included). You will be timed!)

- Title, Authors
- (15%) Motivation: Introduce your question and why the question is interesting. Explain what has been done before in this space. Describe your overall plan. Provide a summary of your results.
- (15%) Data: Describe in detail the data that you are using, including the source(s) of the data and relevant statistics.
- (15%) Modeling: Describe in detail the models (baseline + improvement over baseline) that you use in your approach.
- (30%) Experiments: Provide insight into the effect of different hyperparameter choices. Please include tables, figures, graphs to illustrate your experiments.
- (10%) Conclusions: Summarize the key results, what has been learned, and avenues for future work.
- (15%) Code submission: Provide link to your GitHub repo. The code should be well commented and organized.
- Contributions: Specify the contributions of each author (e.g., data processing, algorithm implementation, slides etc.).

- **Final Report**
    - You should have a very detailed README file in the repo:
        - The README should cover how would you run or read your repo.
        - Use proper Markdown and links to the codespace
    - Try breaking down your code into different processes instead of a long document
        - If possible, think about how to productionalize the code.
        - Any team that provides a functional pipeline to run the training and testing outside of the notebook will be rewarded with 10% of Extra Credit.
    - Code comments are very valuable - code as a Machine Learning Engineer and less like a Data Scientist
    - Data should be hosted outside of Git
        - So the notebook/code should explicitly mention how to get the data if we want to reproduce the results.

- Grade Breakdown
    - Baseline: 25%
    - Final Presentation: 40%
    - Code Space: 35%
- As I mentioned in previous lectures, the final grade will be based on final product, so don’t worry too much about the baseline grade.