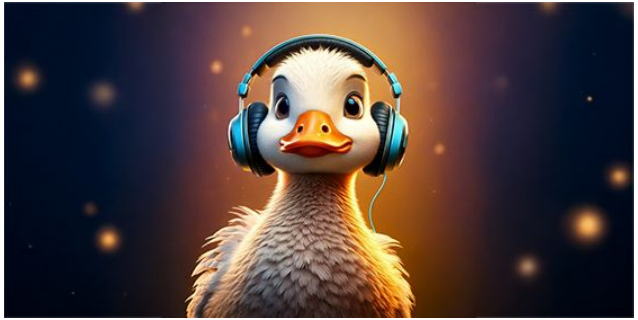
Predict Podcast Listening Time



ICS 435 Final Project

Christian Dela Cruz, Kyler Okuma, Sean Flynn

Background

Our Goal: Predict listening time of a podcast episode.

We were provided with a training dataset containing episode information and a target variable, `Listening_Time_minutes`.
Our objective was to build predictive models to estimate listening time as accurately as possible, measured by Root Mean Squared Error (RMSE) on a separate test set.

Evaluation Metric: Root Mean Squared Error (RMSE)
RMSE was used to measure model performance. RMSE is a standard metric for regression tasks that heavily penalizes larger errors.

$$\text{RMSE} = \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \right)^{\frac{1}{2}}$$

Novelty

- New application area
    - Most ML applications focus on **classification** or **traditional regression**
    - **Predicting user engagement in audio content** has not been explored
- Comparison of ML models
    - Regression
    - MLP
    - LightGBM
    - XGBoost
    - CatBoost
- Simulated real-world ML workflow
    - Feature extraction
    - Tuning
    - Evaluation
    - Validation

Models

- Regression
  - Light
  - Ridge
  - Lasso
- MLP
  - w/ SGD
  - w/ Adam
  - w/ RMSprop
  - w/ Adagrad
  - Ensemble of 5 MLPs
  - Final Optimized MLP (Optuna Parameters)
- Optimized LightGBM
- Stacked Ensemble (All Models Above)
- XGBoost
- CatBoost

Regression

Overview:

- Used Linear Regression, Ridge Regression, and Lasso Regression.
- Assumed a linear relationship between features and listening time.
- Prioritized interpretability over flexibility.

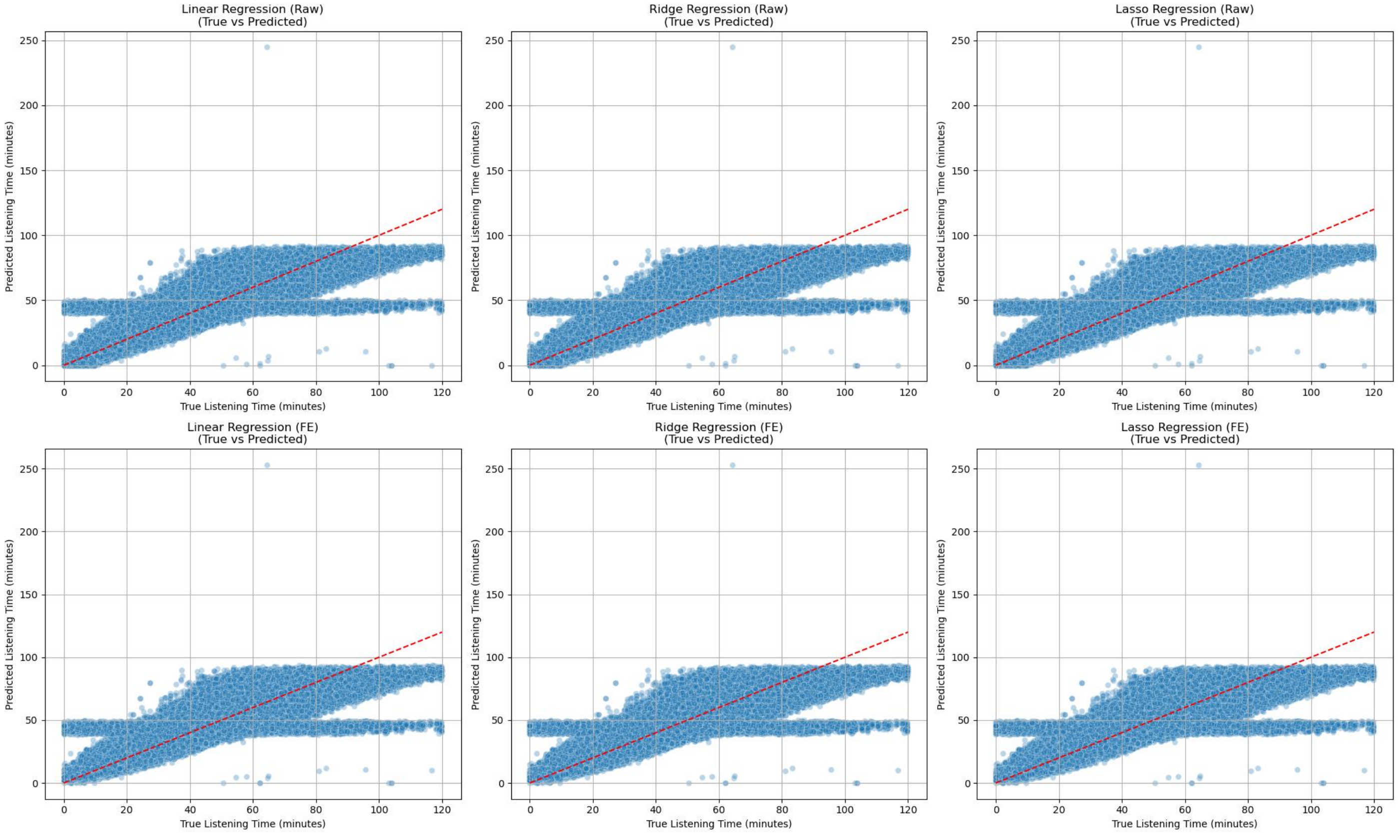
Training Approach:

- Trained separately on RAW and FE datasets.
- GridSearchCV used to tune regularization strength (alpha).
- Categorical features one-hot encoded; missing values imputed.

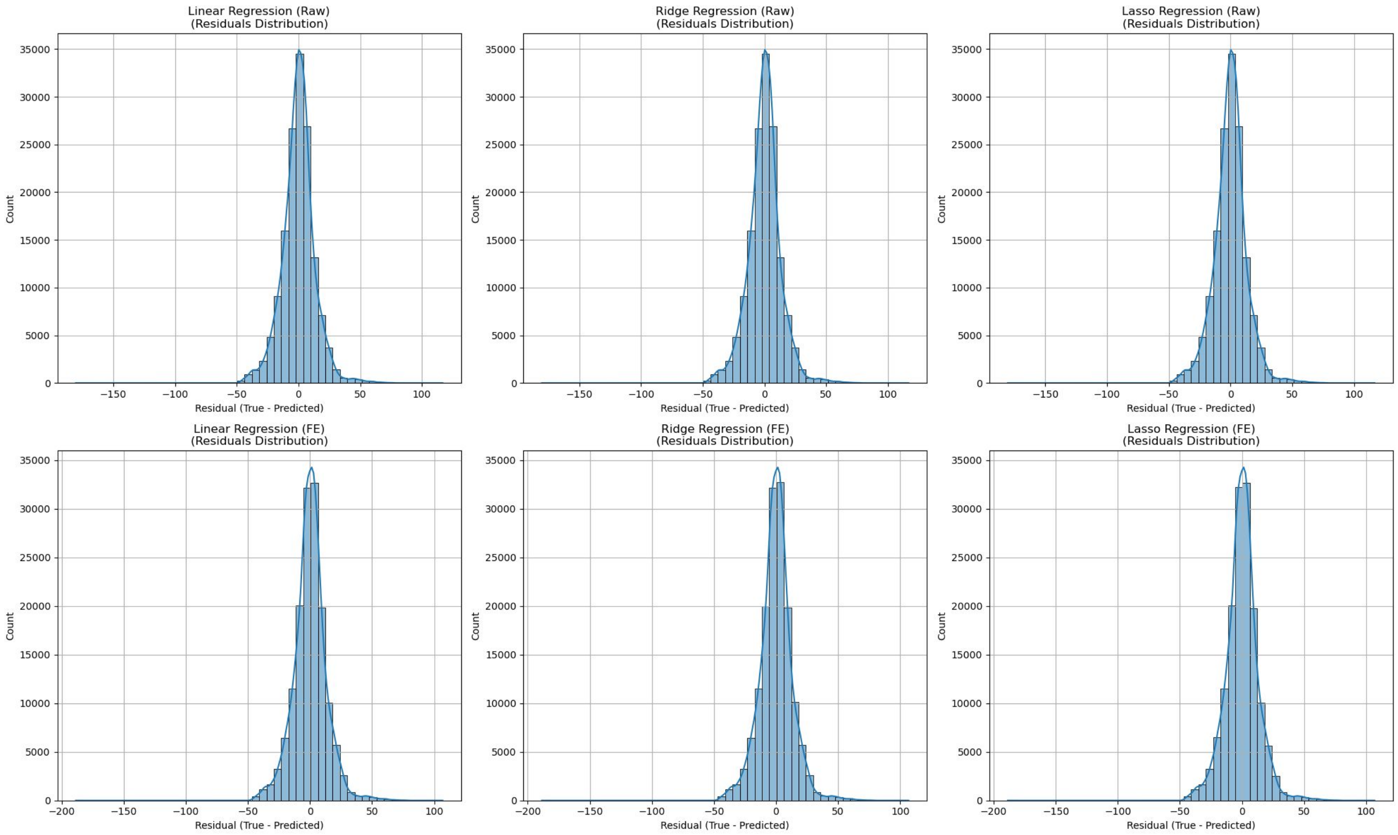
Results:

- Linear Regression (FE): RMSE = 13.296, MAE = 9.742, R^2 = 0.7598
- Ridge Regression (FE): RMSE = 13.295, MAE = 9.741, R^2 = 0.7598
- Lasso Regression (FE): RMSE = 13.295, MAE = 9.741, R^2 = 0.7598

Results - Linear Regression



Results - Linear Regression



MLP

Overview:

- Implemented Multi-Layer Perceptron (MLP) neural networks.
- Aimed to capture nonlinear feature interactions.
- Feedforward architecture capable of approximating complex patterns.

Training Approach:

- Trained on FE dataset with standardized features.
- Tested optimizers: SGD, Adam, RMSprop, Adagrad.
- Adagrad optimizer selected based on best validation results.
- Hyperparameters optimized using Optuna.
- Early stopping used to prevent overfitting.

Results:

- Best Single MLP: RMSE = 13.204, MAE = 9.628, R^2 = 0.7631
- Ensemble of 5 MLPs: RMSE = 13.271, MAE = 9.696, R^2 = 0.7606

Ensemble

Overview:

- Combined outputs of multiple strong models.
- Two ensemble strategies:
    - **Tree-Based Ensemble**: Averaged optimized LightGBM, XGBoost, and CatBoost models.
    - **Full Stacked Ensemble**: Stacked outputs from:
        - Linear Regression (FE)
        - Ridge Regression (FE)
        - Lasso Regression (FE)
        - Best Optuna-tuned MLP
        - Ensemble of 5 MLPs (Adagrad-trained)
        - Best Optuna-tuned LightGBM

Training Approach:

- Optimized each base model separately using GridSearchCV or Optuna.

- Averaged tree predictions for the Tree-Based Ensemble.

- Trained Ridge Regression as meta-learner for Full Stacked Ensemble.

Results:

- Tree-Based Ensemble: RMSE = 12.696, MAE = 9.248, R^2 = 0.7809
- Full Stacked Ensemble: RMSE = 12.931, MAE = 9.382, R^2 = 0.7728

Categorical Boosting

Model Overview

- Common boosting method typically used to handle categorical features
- Known for quick speed and accuracy, as it does not require encoding of categorical features

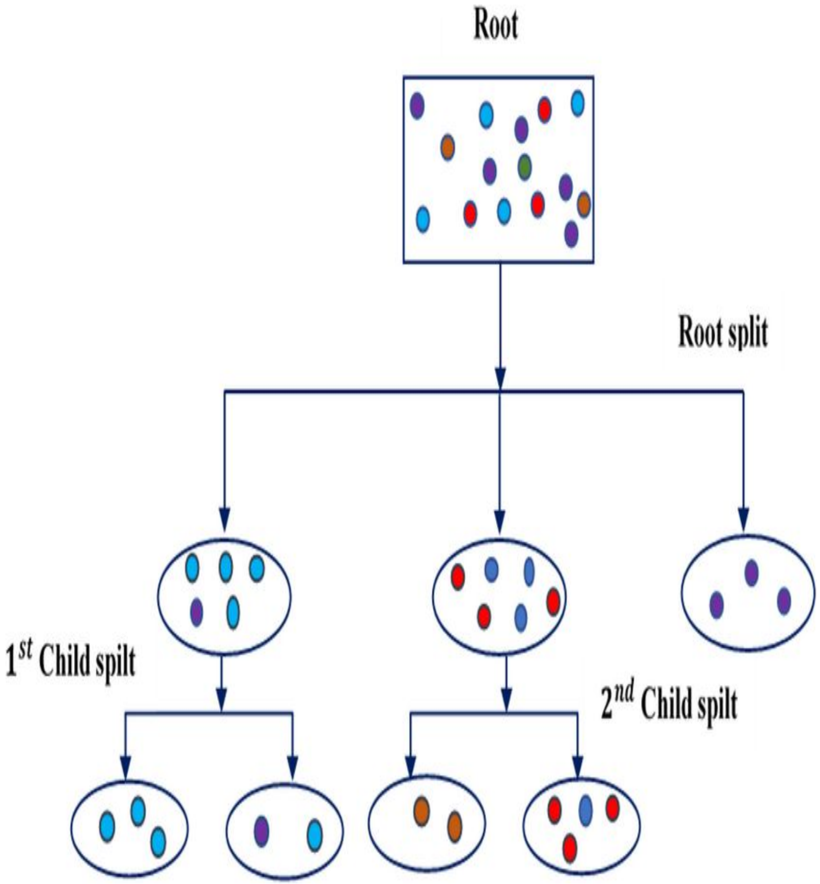


Image Credit: https://www.researchgate.net/figure/Categorical-boosting-CatBoost-algorithm_fig5_380847150

Categorical Boosting (Cont'd)

Training Steps

1) Train with default parameters
2) Perform hyperparameter tuning with optuna
3) 30 trial optimization
4) Train with "optimal" hyperparameters
5) Predict on new data

```
best_catboost_params = {
    'learning_rate': 0.053156533135846354,
    'depth': 11,
    'iterations': 735,
    'l2_leaf_reg': 0.5537806027515884,
    'bagging_temperature': 0.966594578770241,
    'colsample_bylevel': 0.7327316098447535,
    'border_count': 136,
    'random_state': 42,
    'verbose': False
}
```

Optimal hyperparameters trained on model

Categorical Boosting (Cont’d)

Results

- Low RMSE score indicates good performance
- Performed worse than other tree-based models
    - Model’s full strength may not have been utilized, as not *all* of the features were categorical
    - Still followed the common trend throughout the project, with tree-based models performing the best

```
Final Optimized CatBoost Results:
{'Model': 'Final Optimized CatBoost (Optuna Hyperparams)', 'RMSE': 12.997247254066897, 'MAE': 9.4549010868732
6, 'R2': 0.7704238101092351}
```

Final results

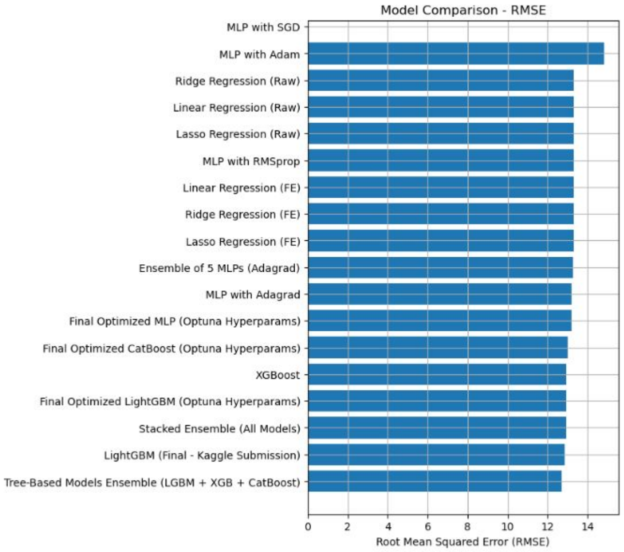
Results - XGBoost Model

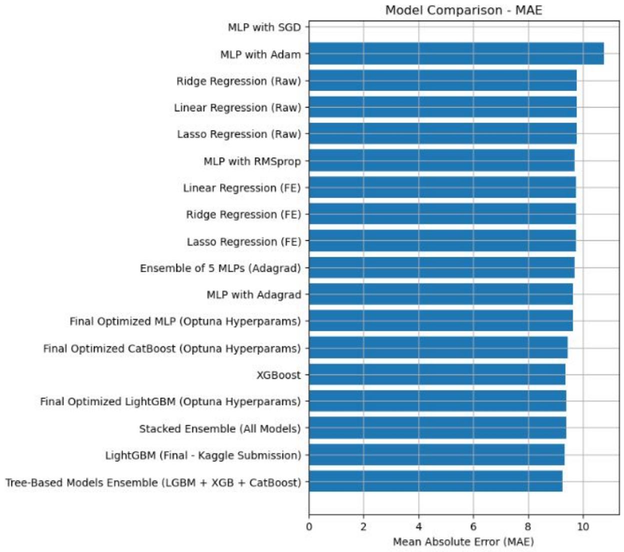
```
[0]	validation_0-rmse:26.09995
[100]	validation_0-rmse:13.09422
[200]	validation_0-rmse:13.05328
[300]	validation_0-rmse:13.02910
[400]	validation_0-rmse:13.01441
[500]	validation_0-rmse:13.00028
[600]	validation_0-rmse:12.99016
[700]	validation_0-rmse:12.98256
[800]	validation_0-rmse:12.97533
[900]	validation_0-rmse:12.96928
[1000]	validation_0-rmse:12.96210
[1100]	validation_0-rmse:12.95727
[1200]	validation_0-rmse:12.95259
[1300]	validation_0-rmse:12.94992
[1400]	validation_0-rmse:12.94535
[1500]	validation_0-rmse:12.94351
[1600]	validation_0-rmse:12.94176
[1700]	validation_0-rmse:12.93899
[1800]	validation_0-rmse:12.93652
[1900]	validation_0-rmse:12.93521
[1999]	validation_0-rmse:12.93239
✅ XGBoost Validation RMSE: 12.93237
```

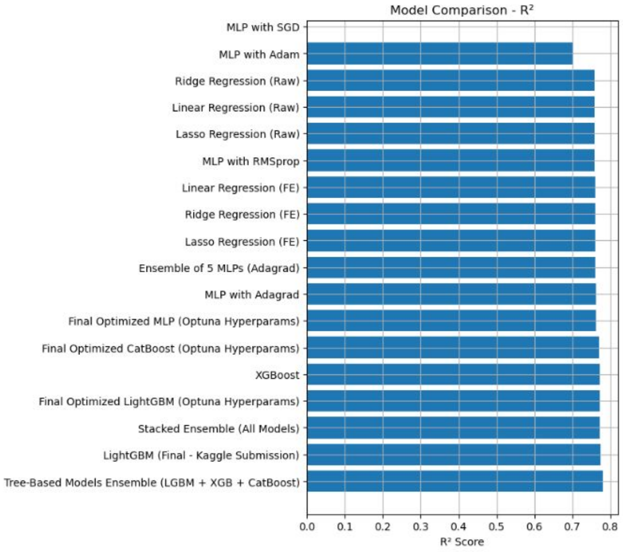
Results - LightGBM Model

```
[100]	training's rmse: 13.001	valid_1's rmse: 13.0542
[200]	training's rmse: 12.8623	valid_1's rmse: 13.016
[300]	training's rmse: 12.7532	valid_1's rmse: 12.9914
[400]	training's rmse: 12.6462	valid_1's rmse: 12.9731
[500]	training's rmse: 12.5645	valid_1's rmse: 12.9577
[600]	training's rmse: 12.4733	valid_1's rmse: 12.9443
[700]	training's rmse: 12.3918	valid_1's rmse: 12.9334
[800]	training's rmse: 12.3185	valid_1's rmse: 12.9239
[900]	training's rmse: 12.2356	valid_1's rmse: 12.9139
[1000]	training's rmse: 12.1584	valid_1's rmse: 12.9049
[1100]	training's rmse: 12.0934	valid_1's rmse: 12.899
[1200]	training's rmse: 12.0259	valid_1's rmse: 12.895
[1300]	training's rmse: 11.9525	valid_1's rmse: 12.886
[1400]	training's rmse: 11.8835	valid_1's rmse: 12.8802
[1500]	training's rmse: 11.8121	valid_1's rmse: 12.8766
[1600]	training's rmse: 11.748	valid_1's rmse: 12.8733
[1700]	training's rmse: 11.6848	valid_1's rmse: 12.8668
[1800]	training's rmse: 11.6277	valid_1's rmse: 12.8632
[1900]	training's rmse: 11.5724	valid_1's rmse: 12.8582
[2000]	training's rmse: 11.5165	valid_1's rmse: 12.8558
Did not meet early stopping. Best iteration is:
[2000]	training's rmse: 11.5165	valid_1's rmse: 12.8558
✅ LightGBM Tuned Validation RMSE: 12.85577
```

Results - Model Comparison





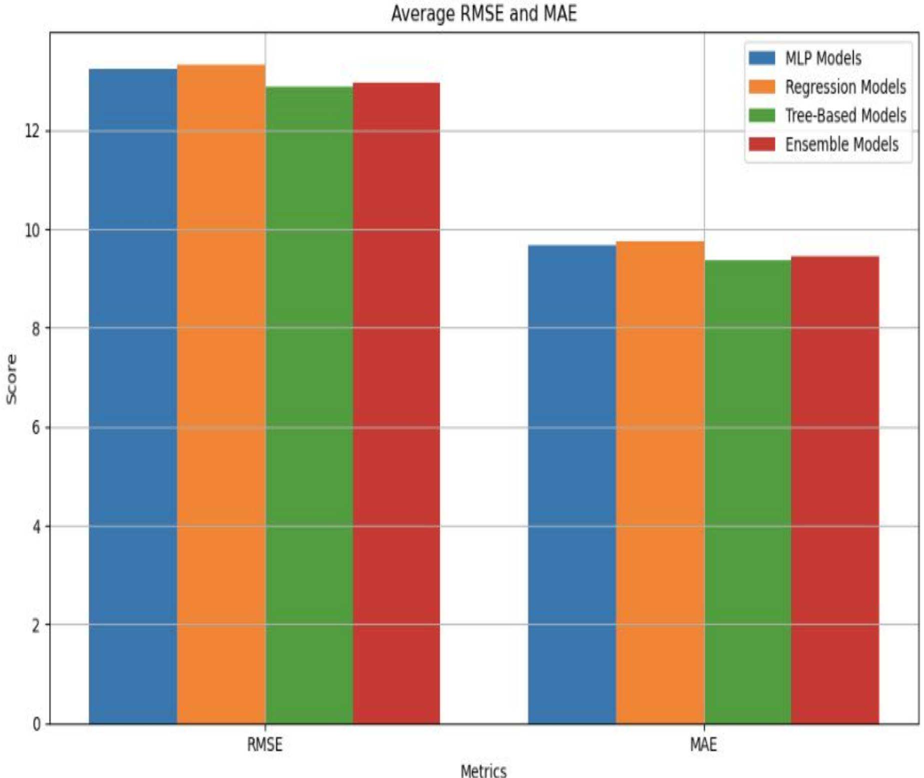


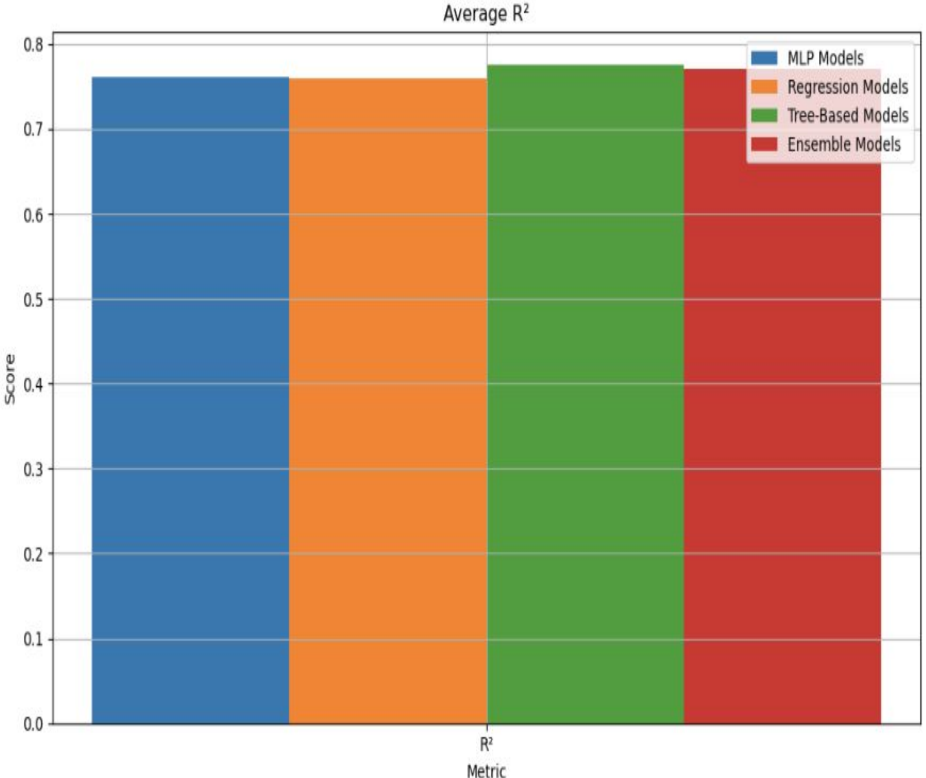
Results

	Model	RMSE	MAE	R2	Type
0	Linear Regression (Raw)	13.319984	9.764266	0.758881	Linear
1	Linear Regression (FE)	13.295618	9.741798	0.759762	Linear
2	Ridge Regression (Raw)	13.320041	9.764298	0.758879	Linear
3	Ridge Regression (FE)	13.295342	9.741233	0.759772	Linear
4	Lasso Regression (Raw)	13.319966	9.764137	0.758882	Linear
5	Lasso Regression (FE)	13.295254	9.741069	0.759775	Linear
6	MLP with SGD	NaN	NaN	NaN	Neural Net
7	MLP with Adam	14.812958	10.779123	0.701800	Neural Net
8	MLP with RMSprop	13.307721	9.686967	0.759325	Neural Net
9	MLP with Adagrad	13.210906	9.647330	0.762814	Neural Net
10	Ensemble of 5 MLPs (Adagrad)	13.271379	9.695907	0.760637	Ensemble
11	Final Optimized MLP (Optuna Hyperparams)	13.203998	9.628222	0.763062	Neural Net
12	Final Optimized LightGBM (Optuna Hyperparams)	12.931977	9.379242	0.772724	Tree-Based
13	Stacked Ensemble (All Models)	12.930729	9.381626	0.772768	Ensemble
14	Final Optimized CatBoost (Optuna Hyperparams)	12.997247	9.454901	0.770424	Tree-Based
15	LightGBM (Final - Kaggle Submission)	12.855765	9.337937	0.775395	Tree-Based
16	XGBoost	12.932367	9.371496	0.772710	Tree-Based
17	Tree-Based Models Ensemble (LGBM + XGB + CatBo...	12.695807	9.247818	0.780949	Ensemble

Data Visualizations: Comparison Between Model Types

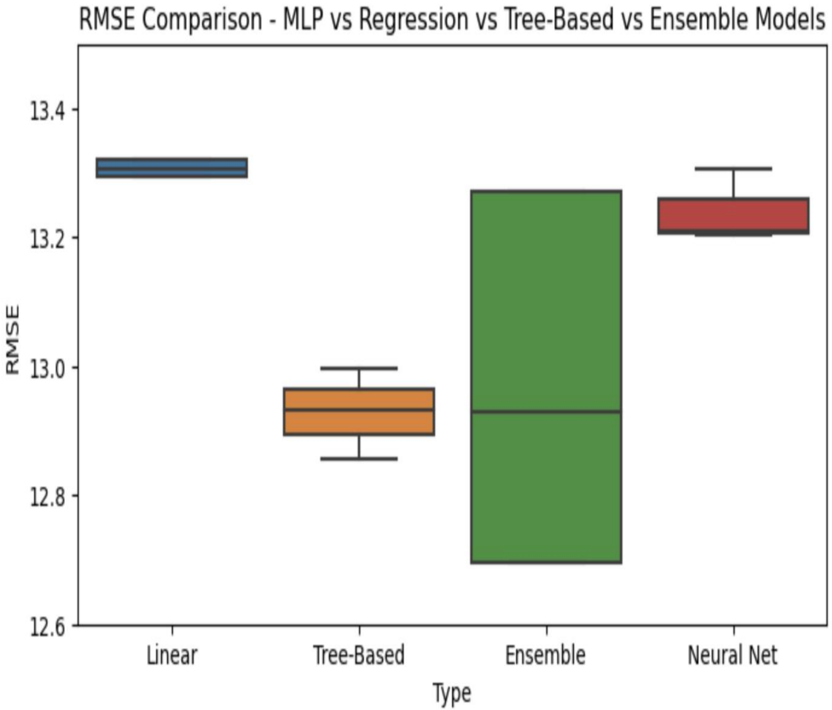
Bar Charts

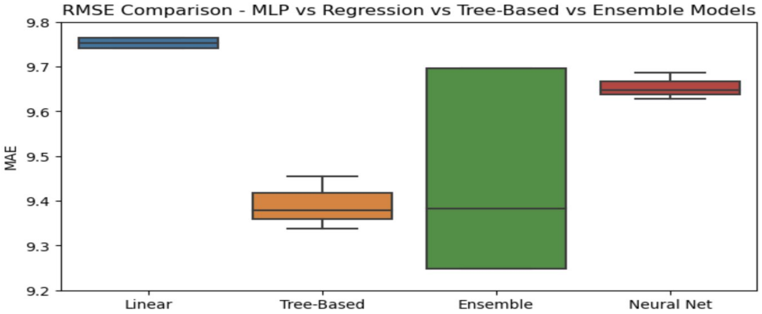


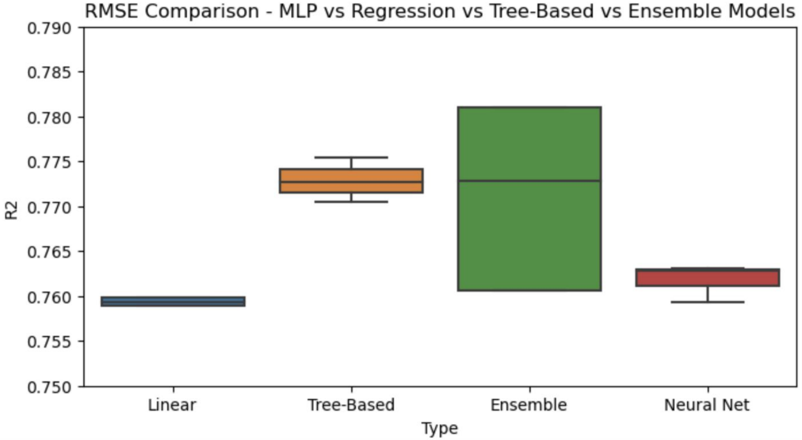


Data Visualizations: Comparison Between Model Types (Cont'd)

Box Charts

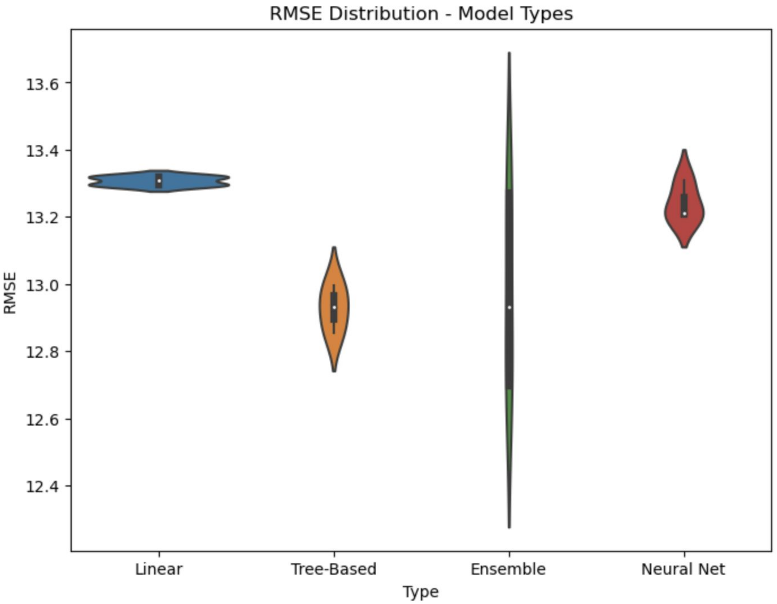


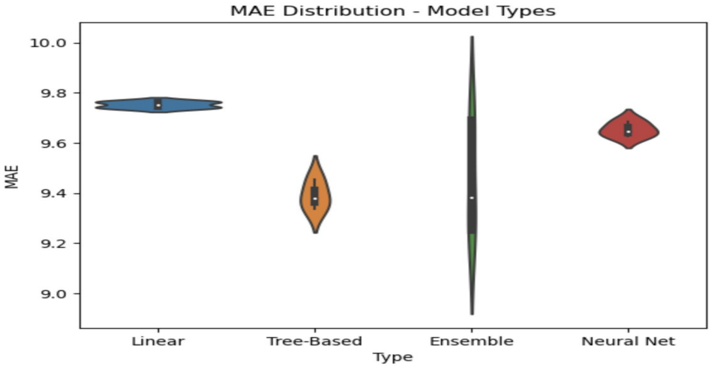


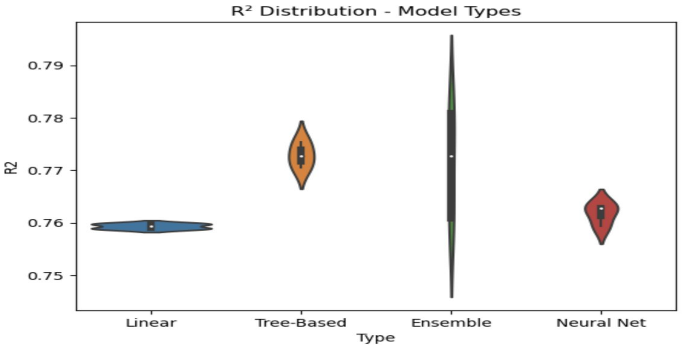


Data Visualizations: Comparison Between Model Types (Cont’d)

Violin Charts



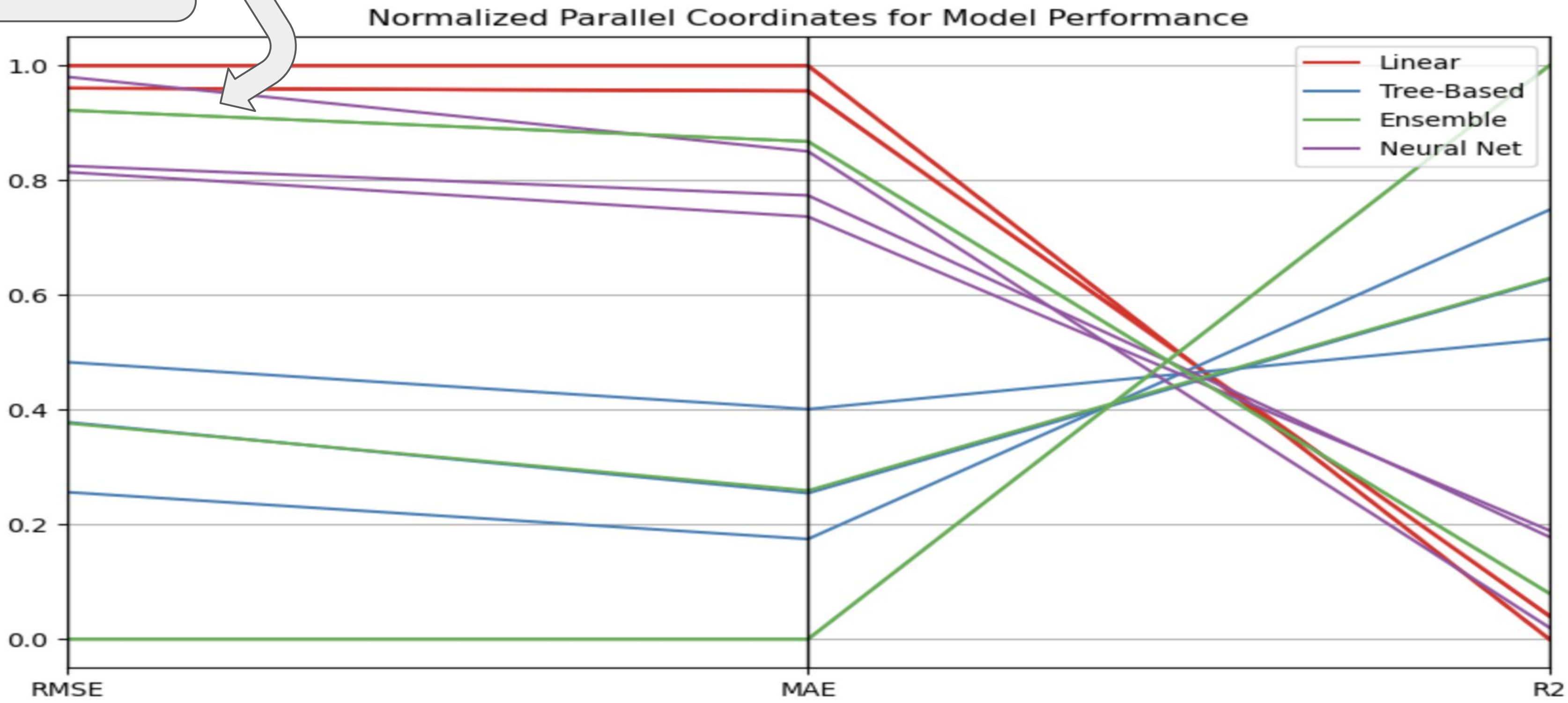




Data Visualizations: Comparison Between Model Types (Cont'd)

Outlier!

Parallel Coordinates



Data Visualizations: Comparison Between Model Types (Cont'd)

- Overall, the tree based models and the ensembles performed the best
  - Lowest RMSE/MAE scores and highest R^2 score
- Ensemble methods had the highest distribution among the data, and linear models had the lowest distribution among the data
- Parallel coordinate chart reveals an inverse relationship between the mean squared error (RMSE/MAE) metrics and R^2 score
- Higher variance among the ensemble methods could imply sensitivity to outliers, possibly explaining why they performed slightly worse than the tree-based models

Conclusion

- LightGBM proved to be a strong model
  - Efficiency, support for categorical features, robustness to missing data - great for the dataset
  - Feature engineering had significant positive impact on performance
- Future improvements:
  - Cross validation
  - Deeper hyperparameter optimization
  - Ensemble with other models such as XGBoost or CatBoost