# Road Accident Risk Prediction

**Benediktus Sashenka (G2505353G)**
AI6102 Machine Learning Methodologies and Applications

## Abstract

According to Singapore Police Force, there are 7,049 road accident cases resulting in injuries and 142 cases of fatal accidents in 2024 in Singapore [1]. Road accidents are an inherent risk for vehicle drivers, cyclists, or even pedestrians around the roadway areas. While risk is unavoidable, there are some risk contributing factors such as internal factors (e.g. speeding, drunk-driving) and external factors (e.g. road curvature, weather). For this project, we aim to train a model to predict the likelihood of accidents based on external risk factors i.e., road-condition features.

## 1 Introduction

Given the road accident risk problem, we first define the outline of this project, which consists of the background context settings, possible challenges that we may face, and relevant approaches to tackle the challenges.

### 1.1 Background

This problem is part of a kaggle competition: Predicting Road Accident Risk [2]. The competition consists of 518k data points in a tabular training dataset with 12 features of road conditions such as `road_type`, `num_lanes`, `curvature`, etc., and the `accident_risk` as the ground truth. The goal of the competition is to predict `accident_risk` for unlabeled 172k data points of testing dataset. The evaluation metric used is root mean-squared error (RMSE), and the final standings in the competition will be evaluated by RMSE score of 80% of the testing dataset prediction. We may select up to 2 submissions to be evaluated to the final leaderboard score.

### 1.2 Challenges

The given dataset has 12 features, consisting of categorical features and both discrete and continuous numerical features. The features may have nonlinear correlation to the ground truth. The competition is scored based on testing dataset RMSE, thus we prefer models that not only perform well on the training dataset but also generalize well to unseen data.

### 1.3 Methodologies

Given the problem challenges, we break down the approach into data preparation and data modelling. The data preparation process includes feature engineering to give additional insights to the data, then we perform encodings and normalization to handle various feature types. In the data modelling stage, we fit the data into several models: Linear Regression, K-Nearest Neighbor, Binary Decision Tree, Random Forest, and Gradient Boosting. For each model, we further experiment by searching for the best hyperparameters to get the optimal model for each type. Then, we select the best 2 models to submit into the competition. To reduce the risk of overfitting, we use 5-fold cross-validation to get the cross-validation RMSE score which we use to determine the best model.

## 2 Data Preparation

We start in the data preparation. In this section, we will go through step-by-step preparation process to prepare the raw data into ready-to-train datasets that can be use for modelling.

### 2.1 Data Exploration

We should start with exploring all the features that are present in the dataset. Figure 1 shows the distributions of each feature.

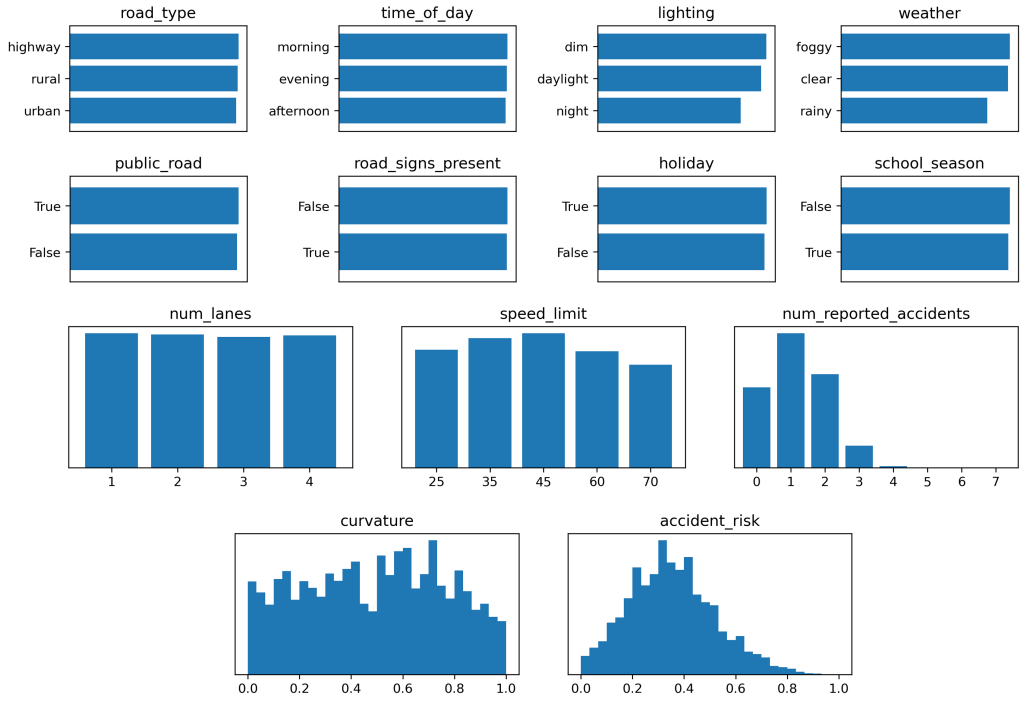



Figure 1: Training dataset feature distributions.

From this observation, the training dataset consists of:

- 8 categorical features
  - 4 non-binary: `road_type`, `time_of_date`, `lighting`, `weather`
  - 4 binary: `public_road`, `road_signs_present`, `holiday`, `school_season`
- 5 numerical features
  - 3 discrete: `num_lanes`, `speed_limit`, `num_reported_accidents`
  - 2 continuous: `curvature`, `accident_risk` (ground truth)

The distributions shows no presence of null values or obvious outliers to be handled. All features appear to be close to a uniform distribution except `num_reported_accidents` and `accident_risk`. With the distribution of the features known, we derive the full plan of data preparation: feature engineering to add meaningful relationship to the data, dealing with categorical values with encodings, and normalize the data.

## 2.2 Feature Engineering

In this section, we observe the feature relationship with the ground truth, i.e. `accident_risk`. Because we are dealing with mostly discrete and categorical features, we set aside `curvature` for now and use boxplot to see the `accident_risk` distributions for each distinct feature values, shown in Figure 2. Some of the features have interesting properties and we construct 3 new features that might add additional data value to the original features.

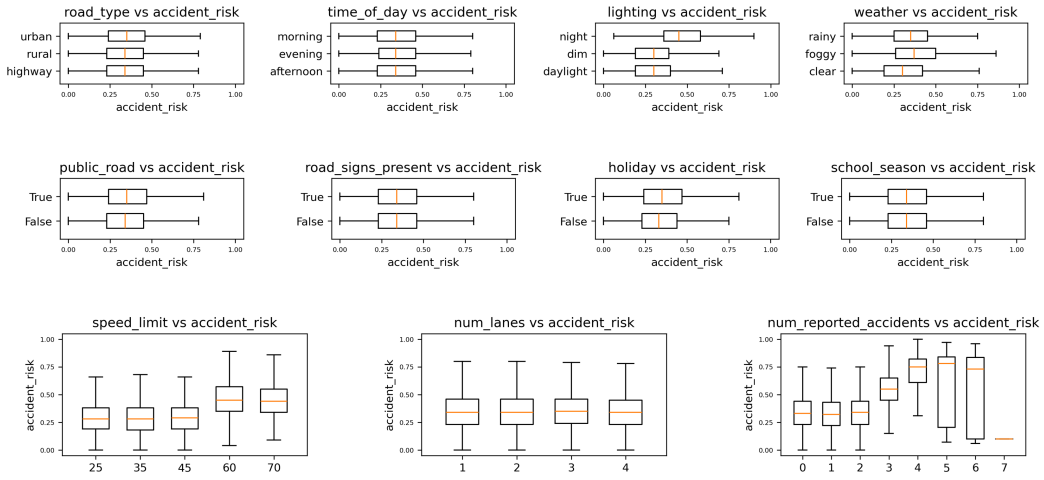



Figure 2: `accident_risk` distributions in respect to discrete and categorical features.

Firstly, we observed that when `speed_limit` is above certain threshold, the `accident_risk` distribution rise significantly. It makes sense for roads with high speed limit have higher risk of accidents. For this relationship, we create a new binary feature:

$$\texttt{speed\_limit\_high} := \texttt{speed\_limit} \geq 60$$

Secondly, `lighting` and `weather` affect driver visibility on the roads, and lower visibility can lead to higher `accident_risk`. We combine these two categorical features to a single discrete numerical feature `light_level`. This feature will benefit the model as it introduces the order of its values. We define the weightage of the 2 feature by the effect of the feature value, i.e. `lighting = night` imposed the greatest shift hence the biggest weightage in the new feature. The value of the feature is defined as follows:

| `light_level` := | | `weather` | | |
|---|---|---|---|---|
| | | `foggy` | `rainy` | `clear` |
| `lighting` | `night` | 0.0 | 0.2 | 0.4 |
| | `dim` | 0.6 | 0.8 | 1.0 |
| | `daylight` | 0.6 | 0.8 | 1.0 |

Finally, there are a sigmoid-like relationship with `num_reported_accidents` and `accident_risk` median. We attempt to linearize the feature by defining:

$$\texttt{num\_reported\_accidents\_sigmoid} := \frac{1}{1 + \exp(3 - \texttt{num\_reported\_accidents})}$$

which maps the original to its sigmoid with center $\texttt{num\_reported\_accidents} = 3$. The mapping itself does not change the discretization of the original feature but distance-based method such as Linear Regression and K-Nearest Neighbor might benefit from this feature.

### 2.3 Categorical Encoding

Binary categorical features can be handled easily with binary encodings. Non-binary categorical features (NBCFs) on the other hand must be handled with care. All NBCFs in the dataset have 3 distinct values and distributed equally as seen in Figure 1. NBCFs that may have ordinal properties such as `lighting` and `weather` has been captured in `light_level`, and for `road_type` and `time_of_day` do not appear to have any particular value that outweights the others, i.e. the 3 values are equally important. For this reason, we choose to do one-hot encoding to all 4 NBCFs, replacing 4 categorical features to 12 binary features.

### 2.4 Data Processing

Now that we have dealt with categorical encoding, we want to normalize all of our features especially for Linear Regression and K-Nearest Neighbor. All categorical features already in binary form after encoding. For the numerical features, there are no clear outliers and the distribution is either close to uniform or Gaussian shape. Hence, we use min-max scaling using range $[0, 1]$ to normalize the data. Within this step, we already obtained the normalized dataset with 23 features.

### 2.5 Principal Component Analysis

Additionally, we might try to reduce the data dimensionality to improve training time and reduce noise. We can also perform principal component analysis (PCA) to capture the highest variance of the data using minimum principal components (PCs). From Table 1, 1 PC can only explain $10.9\%$ of the data, 7 PCs needed to explain half the data and 15 PCs can explain most of the data with $95.6\%$ variance preserved out of 23 features of the original normalized data. This suggests that the dataset features are relatively independent, with most contributing meaningful variance, i.e. low redundancy. Thus, we will be using PCs only for K-Nearest Neighbor which benefits greatly from reduced dimensionality.

Table 1: Cumulative preserved variance of $n$ principal components.

| $n$ | Preserved Variance | $n$ | Preserved Variance |
|---|---|---|---|
| 1 | 0.109 | 10 | 0.736 |
| 2 | 0.199 | 11 | 0.782 |
| 3 | 0.278 | 12 | 0.826 |
| 4 | 0.348 | 13 | 0.870 |
| 5 | 0.415 | 14 | 0.913 |
| 6 | 0.480 | 15 | 0.956 |
| 7 | 0.545 | 16 | 0.994 |
| 8 | 0.610 | 17 | 0.998 |
| 9 | 0.675 | $\geq 18$ | 1.000 |

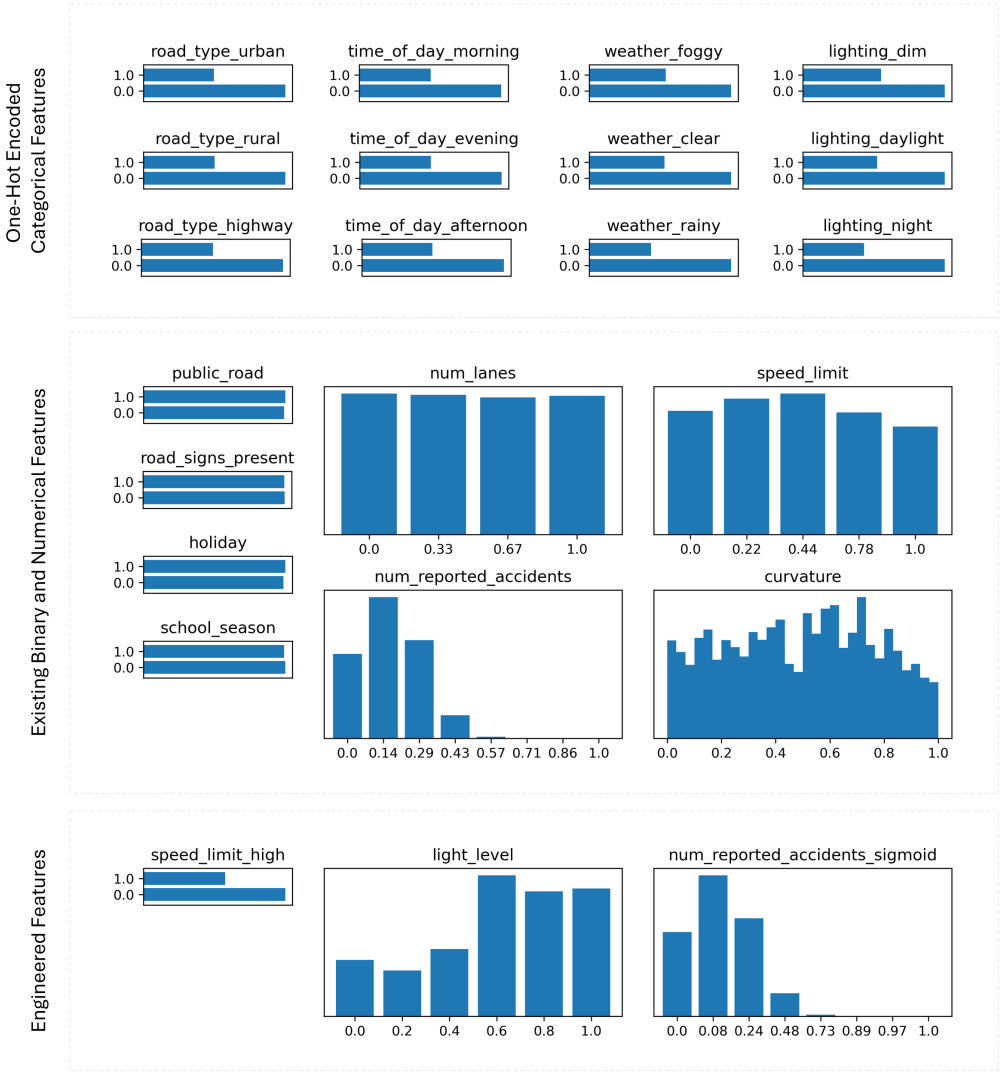



Figure 3: Distribution of processed training dataset features.

## 3 Data Modelling

To summarize the data preparation process, we have successfully created 3 new features, apply one-hot encoding, and scale all features. The resulting ready-to-train dataset consists of 23 features as shown in Figure 3. Now we may begin the modelling process and later compare the performance of each model using this prepared dataset.

### 3.1 Linear Regression

We start with a simple linear regression model. The 5-fold cross-validation for the unregularized linear regression is $0.05916$. Applying the ridge regularization terms does not yield significant improvement to the base unregularized model. Given most of our features do not have strong linear correlation to the target, we use this value as a lower bound benchmark and we expect other models that can utilize the nonlinearity will perform better.

### 3.2 K-Nearest Neighbor

To capture nonlinear correlation of the data, we attempt to model the data using KNN. The model have parameters $k$ number of neighbors and weights option to take the mean of nearest $k$ neighbors (either uniform or L2 norm distance). We fit the model and compare the cross-validation score for both parameters. From Figure 4 we observed uniform weights produce lower RMSE than distance weights for $k \leq 18$. The optimal RMSE is $0.06444$ with $k = 10$ and uniform weightage, which is worse than linear regression because the most of our features are discrete and greatly affected neighbors' distance calculation.

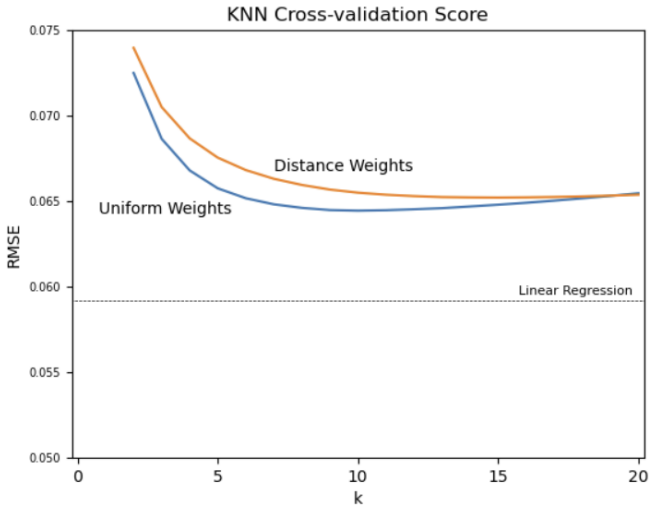



Figure 4: Uniform and distance weighted KNN cross-validation score.

We already perform PCA on the data and we can use the reduced-dimensionality PCs to the KNN model. In Figure 5, using PCs much more effective and stable as well (for $k \leq 50$, we observed the RMSE score converges as $k$ increases). PCA also mitigates the discrete nature of the features, making KNN distance calculation more reliable. For uniform weights KNN, we have the best RMSE score when using 5 PCs. To compare its performance with distance weights counterpart, Figure 6 shows the RMSE difference of uniform weights and distance weights for each parameter $k$. We observed that the difference is always negative for sampled PCs, means that uniform weights always perform better. Hence we choose uniform weights KNN using 5 principal components with $k = 50$, yielding $0.05767$ cross-validation score, an improvement from linear regression model.

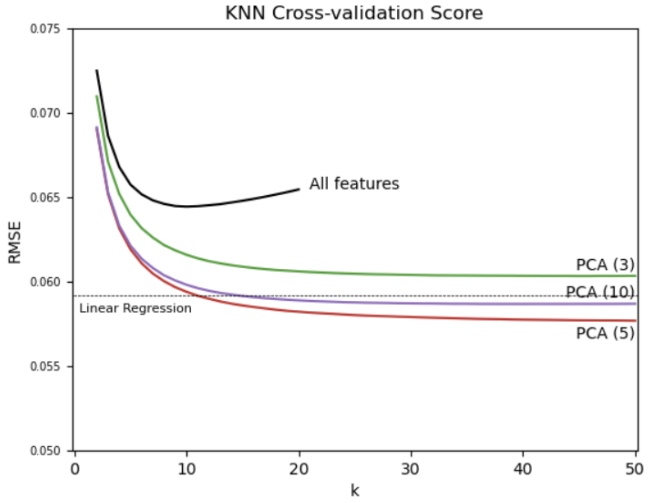



Figure 5: Uniform weight KNN using all features and principal components cross-validation score.

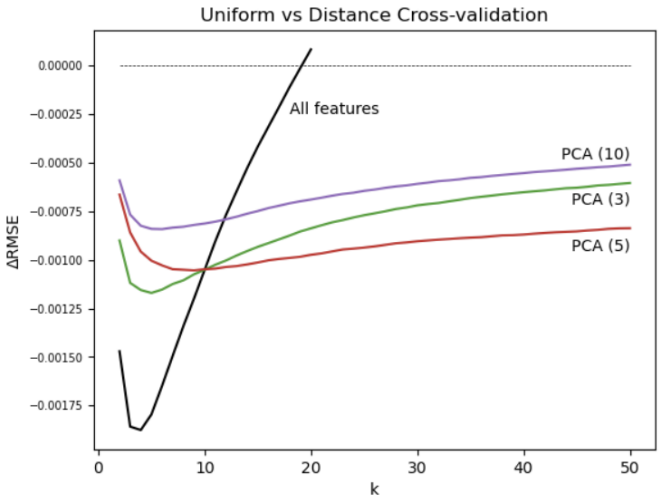



Figure 6: Uniform and distance cross-validation difference for all features and principal components, $\Delta\text{RMSE} = \text{RMSE}_{\text{uniform}} - \text{RMSE}_{\text{weight}}$ (of KNN cross-validation score).

### 3.3 Binary Decision Tree

Another model that can capture nonlinearity of the data is Binary Decision Tree. Decision trees naturally handle categorical data, which is advantageous compared to KNN. Given we have many categorical fields in the dataset, BDTs might be preffered. But as the model complexity increases, the model has more hyperparameters to be optimized. We experiment on hyperparameters: `max_depth`, `min_samples_split`, and `min_samples_leaf`. The main hyperparameter to set the complexity of the tree is `max_depth`, and `min_samples_split` and `min_samples_leaf` acts more like a regularization hyperparameter to prevent overfitting. We find the optimal tree with the best cross-validation score using grid search, i.e. all possible combinations of candidate parameter values in Table 2.

Table 2: Experimented hyperparameter values for Binary Decision Tree.

| Parameter | Default Value | Candidate Values |
|---|---|---|
| `max_depth` | – | $\{6, 7, \ldots, 15\}$ |
| `min_samples_split` | 2 | $\{2, 5, 10, 50, 100, 500, 1000\}$ |
| `min_samples_leaf` | 1 | $\{1, 2, 5, 10\}$ |

Table 3: Optimal `min_samples_split` and `min_samples_leaf` for each `max_depth`.

| `max_depth` | `min_samples_split` | `min_samples_leaf` | RMSE |
|---|---|---|---|
| 6 | 2 | 2 | 0.05836 |
| 7 | 50 | 2 | 0.05702 |
| 8 | 2 | 2 | 0.05652 |
| 9 | 100 | 2 | 0.05636 |
| 10 | 100 | 2 | 0.05630 |
| 11 | 500 | 5 | 0.05628 |
| 12 | 500 | 5 | 0.05630 |
| 13 | 1000 | 5 | 0.05632 |
| 14 | 1000 | 5 | 0.05632 |
| 15 | 1000 | 5 | 0.05632 |

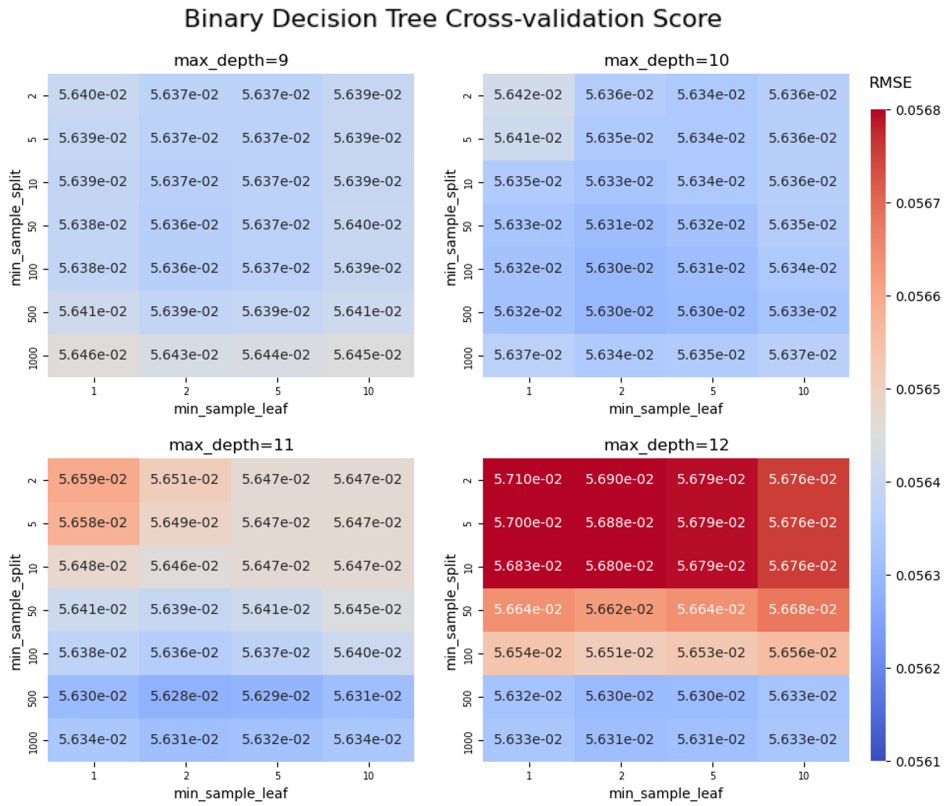



Figure 7: BDT cross-validation scores for $9 \leq$ `max_depth` $\leq 12$.

From Table 3 and Figure 7, we observed that the model performance is highly sensitive of hyperparameters. For `max_depth` $< 9$, high RMSE indicates the tree have not yet capture the data patterns, resulting underfitting. While for `max_depth` $> 10$, the model performed better if we increase the parameter `min_samples_split` to prevent high complexity tree from overfitting. The optimal tree is given by `max_depth` $= 11$, `min_samples_split` $= 500$, and `min_samples_leaf` $= 2$, resulting RMSE of $0.05628$.

### 3.4 Random Forest

While a single decision tree can model complex relationships effectively, it is prone to overfitting risk. In the previous section, we limit the risk by adjusting hyperparameter `min_samples_split` and `min_samples_leaf`, but it can be improved more with ensemble techniques such as random forest. Random Forest is an estimator that fits a number of decision tree regressors on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [3].

To search the optimal hyperparameter, we take the similar approach and hyperparameter search as BDT. We know that `max_depth` $= 11$ is optimal for the BDT, so we can narrow down the forest hyperparameter search for that variable. Additionally, we have a new hyperparameter `n_estimators` for the number of independent trees.

Table 4: Experimented hyperparameter values for Random Forest.

| Parameter | Default Value | Candidate Space |
|---|---|---|
| `n_estimators` | 100 | $\{10, 20, 50, 100, 200\}$ |
| `max_depth` | – | $\{9, 10, 11, 12\}$ |
| `min_samples_split` | 2 | $\{2, 5, 10, 50, 100, 500, 1000\}$ |
| `min_samples_leaf` | 1 | $\{1, 2, 5, 10\}$ |

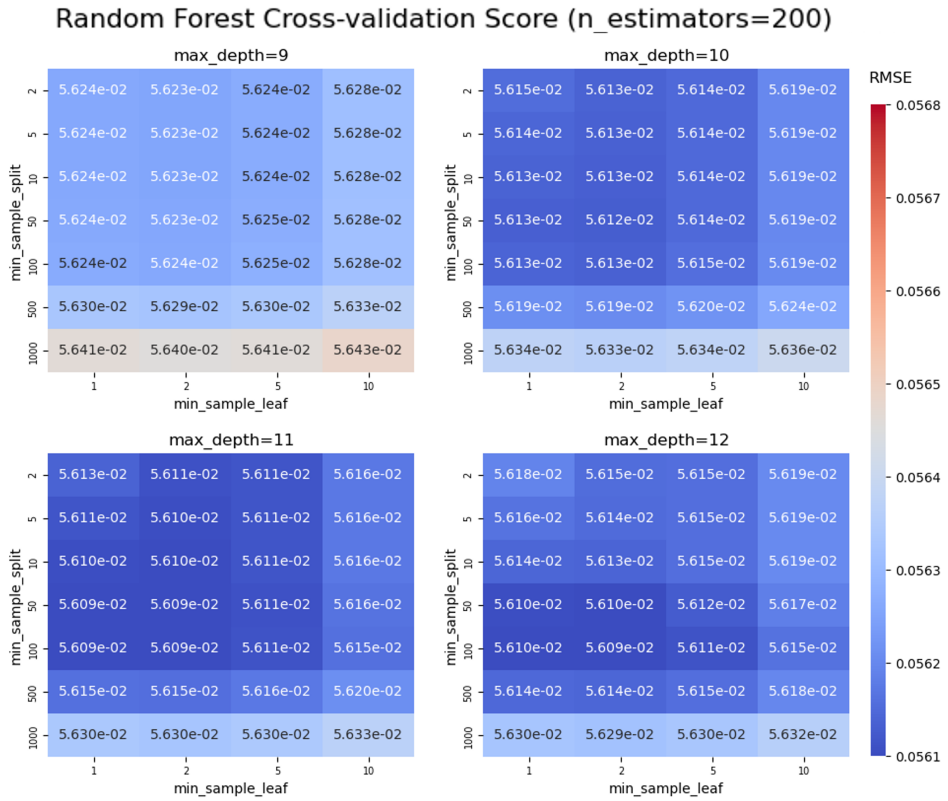



Figure 8: RF cross-validation scores for $9 \leq$ `max_depth` $\leq 12$ and `n_estimators` $= 200$.

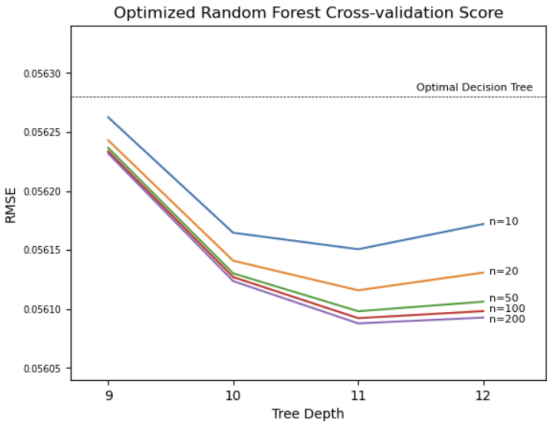



Figure 9: Optimal Random Forest hyperparameters for each `max_depth` and `n_estimators`.

As expected, higher `n_estimators` resulting in better cross-validation score, as seen in Figure 4. We just use `n_estimators` $= 200$ as the RMSE already nearly converges. Alongside the better cross-validation score, the model also behaving more stable especially with higher complexity trees (e.g. `max_depth` $= 12$), as the average estimates reduces the chance of overfitting unlike a single decision tree and it produces more general and reliable solution. Thus the optimal forest have cross-validation score of $0.05609$ with hyperparameters:

- `n_estimators` $= 200$
- `max_depth` $= 11$
- `min_samples_split` $= 50$
- `min_samples_leaf` $= 2$

### 3.5 Gradient Boosting

Another ensemble techniques is Gradient Boosting Decision Tree (GBDT), in which with each iterations a decision tree learns from the negative loss gradients from the previous ones. GBDT is a widely-used because of its efficiency and accuracy. We are using one of the implementations of GBDT algorithm, LightGBM, which scales well from large number of data instances and features [4].

LightGBM offers many available hyperparameters [5]. Instead of using grid search to find the optimal hyperparameter, we use `Optuna` as a hyperparameter search framework that uses efficient sampling and pruning mechanisms to find the optimal parameter combination across the possible parameter space[6]. Each parameter combination sampling in `Optuna` called a "study", and we collect the cross-validation score for each 1000 studies in the parameter space shown in Table 5.

Table 5: Sampled hyperparameter space for LightGBM.

| Type | Parameter | Default Value | Candidate Space |
|---|---|---|---|
| Structure Constraints | `n_estimators` | 100 | $\{500, 501, \ldots, 2000\}$ |
| | `learning_rate` | 0.1 | $[0.01, 0.2]$ |
| | `num_leaves` | 31 | $\{16, 17, \ldots, 1023\}$ |
| | `max_depth` | – | $\{5, 6, \ldots, 25\}$ |
| Sampling Parameters | `min_child_samples` | 20 | $\{5, 6, \ldots, 100\}$ |
| | `min_split_gain` | 0 | $[0.0, 1.0]$ |
| | `subsample` | 1 | $[0.6, 1.0]$ |
| | `subsample_freq` | 0 | $[1, 10]$ |
| | `colsample_bytree` | 1 | $[0.6, 1.0]$ |
| Regularizations | `reg_alpha` | 0 | $[0.0, 1.0]$ |
| | `reg_lambda` | 0 | $[0.0, 1.0]$ |

Table 6: Top 5 hyperparameter studies of LightGBM.

| Study ID | **611 (best)** | 764 | 578 | 583 | 815 |
|---|---|---|---|---|---|
| `n_estimators` | 1855 | 1813 | 1870 | 1921 | 1911 |
| `learning_rate` | 0.006 | 0.006 | 0.006 | 0.006 | 0.005 |
| `num_leaves` | 170 | 168 | 145 | 160 | 194 |
| `max_depth` | 19 | 17 | 20 | 19 | 19 |
| `min_child_samples` | 7 | 6 | 5 | 5 | 5 |
| `min_split_gain` | $8.81 \times 10^{-4}$ | $7.51 \times 10^{-4}$ | $1.11 \times 10^{-3}$ | $9.64 \times 10^{-4}$ | $7.22 \times 10^{-4}$ |
| `subsample` | 0.889 | 0.905 | 0.876 | 0.881 | 0.877 |
| `subsample_freq` | 10 | 10 | 10 | 10 | 10 |
| `colsample_bytree` | 0.661 | 0.676 | 0.674 | 0.670 | 0.692 |
| `reg_alpha` | $8.00 \times 10^{-6}$ | $6.66 \times 10^{-4}$ | $9.58 \times 10^{-4}$ | $1.15 \times 10^{-3}$ | $5.70 \times 10^{-4}$ |
| `reg_lambda` | $3.17 \times 10^{-5}$ | $3.86 \times 10^{-2}$ | $2.42 \times 10^{-8}$ | $2.96 \times 10^{-8}$ | $2.21 \times 10^{-8}$ |
| `RMSE` | 0.055960 | 0.055961 | 0.055962 | 0.055962 | 0.055962 |

From Table 6, we observed the structure constraints sampling parameters in the top 5 studies relatively converges in RMSE, having similar performances. The top 5 studies also shows similar structure constraints and sampling parameters, which means the studies can capture the optimal hyperparameter for the lightGBM. Furthermore, we observed varying L1 and L2 regularization terms, implying the model is not sensitive to these parameters. We take the best study as the optimal LightGBM with cross-validation score $0.05596$.

## 4 Final Model

Now we have the optimized cross-validation score for each model through experimenting possible hyperparameters. The summarized result is given in the Table 7 below.

Table 7: Summary of model cross-validation scores.

| Model | Cross-validation RMSE |
|---|---|
| Linear Regression | 0.05916 |
| K-Nearest Neighbor | 0.05767 |
| Binary Decision Tree | 0.05628 |
| Random Forest | 0.05609 |
| Gradient Boosting (LightGBM) | 0.05596 |

For the competition submission, we may choose the 2 best model to be scored in the leaderboard. We choose Random Forest and LightGBM to be submitted into the competition. The final score for both models against full testing data shown in Table 8.

Table 8: Random Forest and LightGBM final scores.

| RMSE | Cross-validation | Public Score (20% of testing data) | Private Score (Final) (the other 80% of testing data) |
|---|---|---|---|
| Random Forest | 0.05609 | 0.05561 | 0.05587 |
| diff with cross-validation | | −0.00048 | −0.00022 |
| LightGBM | 0.05596 | 0.05551 | 0.05574 |
| diff with cross-validation | | −0.00045 | −0.00022 |

Both models generalize well with unseen testing data. The 5-fold cross-validation prevents models from overfitting, thus maintaining accuracy. LightGBM performed slightly better than the Random Forest against all datasets. The final leaderboard shows this LightGBM model submission placed in the top 13% with RMSE 0.05574.

## 5 Conclusion

More complex model generally gives better accuracy. Models can benefit from feature engineering (e.g. using PCA in KNN) and hyperparameter tuning (e.g. `Optuna` study in LightGBM). Given the nature of the data which is mostly categorical, tree-based methods such as BDT, Random Forest, and LightGBM perform much more effective than linear regression and KNN. Cross-validation prevents models from overfitting and helps model to generalize over unseen data. While choosing the most complex model is preferable in competition settings, we may prefer simpler models in other scenarios where little accuracy score can be sacrificed in tradeoffs with effort and model's explainability.

## References


[1] Singapore Police Force. (2025). Annual road traffic situation 2024. `https://www.police.gov.sg/-/media/SPF/Media-Room/Statistics/Annual-Road-Traffic-Situation-2024/Police-News-Release---Annual-Road-Traffic-Situation-2024.pdf`.

[2] Walter Reade and Elizabeth Park. (2025). Predicting road accident risk.

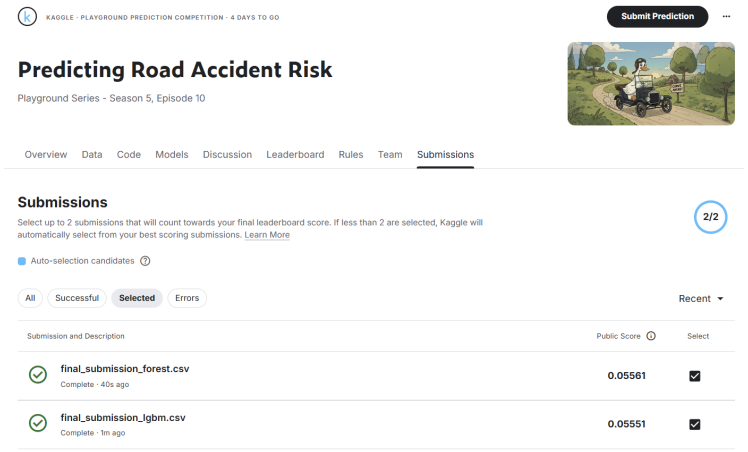
[3] Scikit-learn: Random forest regression api reference. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`.

[4] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

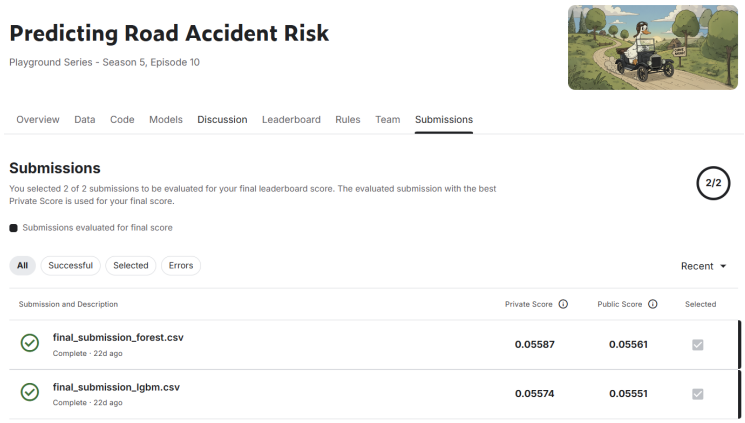
[5] Lightgbm: Parameters tuning. `https://lightgbm.readthedocs.io/en/stable/Parameters-Tuning.html`.

[6] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

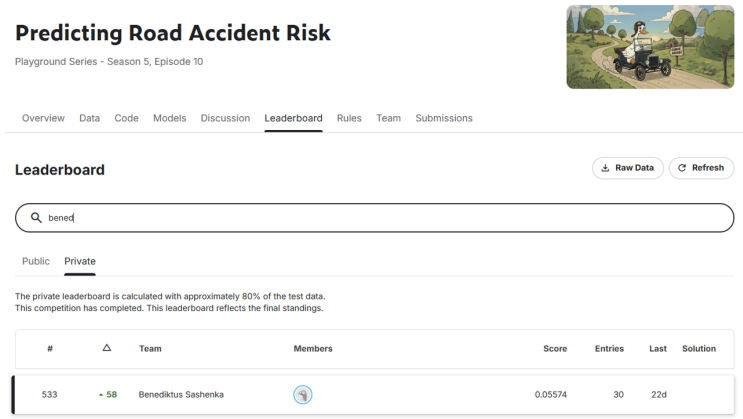
# Appendix: Kaggle competition documentations




Model submissions in competition, with public score.




Model evaluation on end of competition, with private and public score.




Final competition leaderboard entry.