Predictions on California Road Accidents Severity

Xiaoer Hu,
huxiaoer@berkeley.edu

Xiaoyu Yang
xiaoyu_yang@berkeley.edu

Lu Yu
lu.yu97@berkeley.edu

Abstract

Car accidents cause many deaths each year. Accurate predictions help emergency responders to evaluate potential impacts and prepare accordingly. Our project is to predict the severity of car accidents in California using machine learning methods. This paper first explains the data processing method and then analyzes different machine learning models we use for prediction. Different models are used to make predictions. Various hyperparameters and machine learning tricks are further analyzed. The performance of a model is basically evaluated by the accuracy of the prediction. Among the models we use, the decision tree gives the highest accuracy at 0.927.

1. Introduction

Approximately 1.35 million people die each year as a result of road traffic crashes [1]. Road traffic injuries are the leading cause of death for children and young adults aged 5-29 years. It is crucial to predict the severity of accidents, which provides possibilities of effective accident management to decrease injuries and increase traffic safety. More specifically, accurate prediction provides important information for emergency responders to evaluate potential impacts on public safety and prepare accordingly.

2. Dataset

The dataset we start with contains about 3 million traffic accidents that took place in 49 states of the United States from February 2016 to December 2019 [2]. The attributes are summarized in Table 1 below.

Total Attributes	49
Traffic Attributes (12)	id, source, TMC, severity, start_time, end_time, start_lat, start_lng, end_lat, end_lng, distance, and description
Address Attributes (10)	number, street, side (left/right), city, county, state, zip-code, country, timezone, airport_code
Weather Attributes (10)	weather_timestamp, temperature, wind_chill, humidity, pressure, visibility, wind_direction, wind_speed, precipitation, weather_condition
Point of Interest Attributes (13)	amenity, bump, crossing, give-way, junction, no-exit, railway, roundabout, station, stop, traffic calming, traffic signal, turning loop
Period-of-Day (4)	sunrise/sunset, civil twilight, nautical twilight, astronomical twilight

Table 1. The features in the original dataset

We count the accident occurrences in each state, as shown in Figure 1, and realize California state has approximately ¼ of the accidents. Therefore, the accident severity in CA will be analyzed in this report.

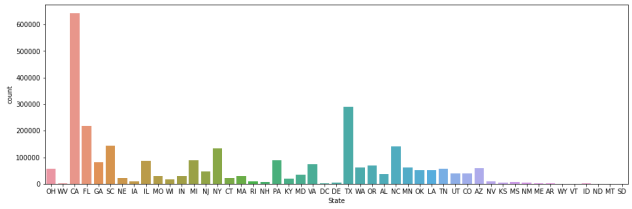

Figure 1. The accident counts in each state.

Based on the accident start time and end time text, we extract the numerical year, month, day, and hour information. Moreover, we create a feature that determines whether the accidents occur during the weekdays or weekends. The accident time duration is calculated for all accidents by subtracting the start time from the end time. We neglect the data that has negative time duration, because obviously there are mistakes while recording the data. For time duration that is out of 3 standard deviations from the median duration, we backfill them with the median.

Wind speed is also an important factor, but in the original dataset, some data is empty for this attribute. We find the wind directions for the empty wind speed data are

either “North” or “Calm”. Therefore, for the empty wind speed data, if its wind direction is “Calm”, we set it to 0, and if its wind direction is “North”, we fill it with the median of the wind speed.

The wind direction feature has more than 20 different descriptions. We first convert them into lower case letters and then create 4 features (“North”, “South”, “East”, “West”) to summarize the wind directions. For example, if the wind direction is Northeast or NE, we will set its North and East features to 1, and keep the other two features 0.

Similarly, for the weather conditions, there are more than 120 different types of descriptions. Based on the lowercase keywords, we summarized the weather conditions into 6 features, as shown in Table 2.

New Features	Keywords
Rainy	rain, drizzle, shower, hail, thunder, storm
Snowy	snow, freez, hail, ice, wintry, sleet
Windy	wind, storm, squall, tornado
Cloudy	overcast, cloud
Fog	haze, fog, smoke, ash, mist, sand, dust
Clear	clear, fair, n/a

Table 2. The keywords for new weather condition features

We also convert other text features into one-hot encoded features, such as Source (Bing/MapQuest), Sunrise_Sunset (day/night), Side (left/right), etc.

After data preprocessing, we have 42 features excluding severity for further analysis.

3. Models

In this section, we analyze three main machine learning models. The advantages and weaknesses of models, hyperparameter tuning and metric selection will be discussed in detail. We also analyze the influence of ensemble learning.

3.1. Decision Tree

A decision tree is a nonlinear machine learning method for classification and regression. In a decision tree, we usually have two node types: internal nodes represent a “test” on feature values, and their branches represent the outcome of the test; leaf nodes represent class labels. Classification rules can be obtained by traversing from root to leaf.

3.1.1 Pre-pruning

A basic decision tree algorithm keeps subdividing tree nodes until every leaf is pure. Sometimes, due to the noise existing in data, it may cause overfitting. Besides, a complete tree can have super considerable depth and tree size. To limit tree size and depth for speed and avoiding overfitting, we need pre-pruning, that is to say, stopping the tree-building process early.

There are several most common stopping conditions, including setting max depth, minimum sample split, and minimum sample leaf. Max depth means when a decision tree arrives at a certain depth, it will stop dividing. The minimum sample split is the minimum sample number in an internal node. If the number of samples in a node is smaller than the minimum sample split, it will stop splitting. The minimum sample leaf means the minimum number of samples required to be at a leaf node.

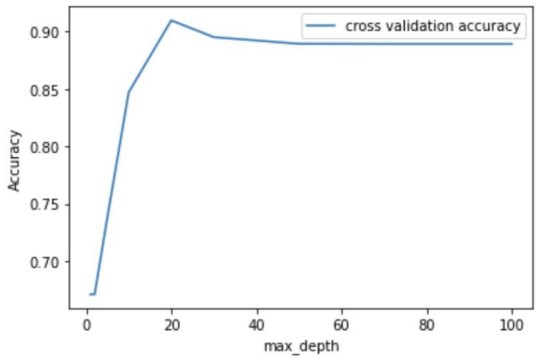

Figure 2. Max depth vs cross validation accuracy of decision tree models.

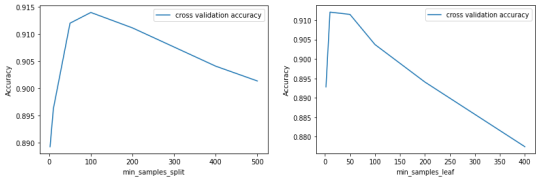

Figure 3. Minimum sample split/minimum sample leaf vs cross validation accuracy of decision tree models.

In our work, we tune these pre-pruning parameters and observe clear rules: with increasing these parameters, the cross-validation accuracy increases at first and then decreases, as shown in Figure 2 and 3. For these three parameters, smaller values mean the tree-building stops earlier, vice versa. If we stop too early, the classifier will get a poor accuracy due to the underfitting. However, if we stop too late, the classifier will fit noise and give a bad performance. The best minimum samples split is about

twice as much as the minimum samples leaf. If we set the minimum sample leaf as a specific value, then the minimum samples split is around twice that value.

3.1.2 Balanced weights

In a basic decision tree, we treat each data sample equally. The basic decision tree works well when the data is balanced. Balanced data means we have similar numbers of samples for each class. However, when the data is imbalanced, we need to do some extra processing on the data. We use a set of weights that is inversely proportional to class frequencies in the data. The results are shown in Figure 4.

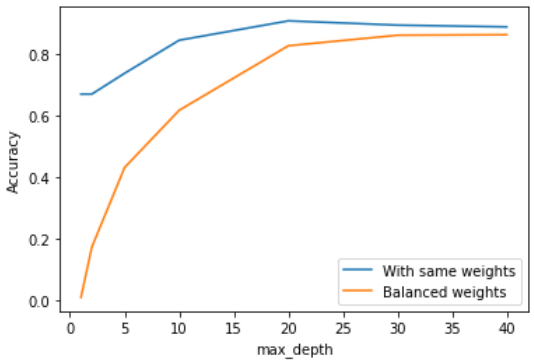

Figure 4. Accuracy of decision tree models vs. maximum depth.

Different from our expectation, the model of balanced weight does not perform better than the normal one. It is because our validation has the same distribution as the imbalanced training data.

3.1.3 Minimal cost-complexity pruning

Minimal cost-complexity pruning is an algorithm used to prune a decision tree to avoid overfitting. This algorithm is more reliable than stopping early. We first build a complete decision tree whose leaves are all pure. Then, we select an α as the complexity parameter. The parameter α can be calculated by $\frac{R(t)-R(T_t)}{|T|-1}$, where $R(T)$ is defined as the misclassification rate of the terminal nodes, $|T|$ is the number of terminal nodes in T and T_t is defined to be a tree whose root is node t [3]. The larger α is, the more impurity decreasing a node can cause.

To implement the pruning, we select a cost complexity threshold. When a branch's minimal α is smaller than the threshold parameter, we prune that branch. During pruning, we first find an α path. This α path returns the effective α in increasing order. As α increases, more branches are pruned. Here we plot the accuracy and depth vs. alpha as shown in Fig. 5 and 6.

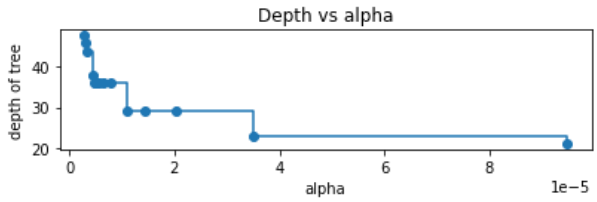

Figure 5. Accuracy of decision tree models vs. α.

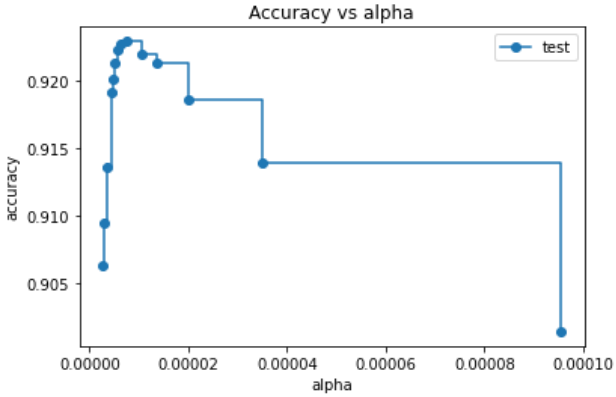

Figure 6. Accuracy of decision tree vs. α.

When the depth is around 30, the best accuracy is 0.927. This accuracy is better than the previous parameter combination. Pruning often works better than stopping early. The reason is that sometimes a split that does not seem to make much progress is followed by a split that makes much progress.

3.2. K-Nearest Neighbor

K-Nearest Neighbor (k-NN) is an instance-based learning algorithm that is effective in both classification and regression tasks [4].

The main idea is that similar things exist in close proximity. In k-NN classifier, we classify test data based on the votes from k nearest neighbors.

3.2.1 Distance Metrics

As k-NN makes predictions based on votes from neighbors, it is important to choose a suitable metric to calculate distances between data points.

We selected three most commonly used distance metrics for evaluation: Euclidean distance, Manhattan distance and Chebyshev distance. Geometry features of metrics are shown in Figure 7.

(a)Euclidean (b)Manhattan (c)Chebyshev

Figure 7. Geometry features of metrics.

Euclidean distance is the most commonly used metric in general k-NN models. It is given by:

$$d_2 : (x, y) \rightarrow \|x - y\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Manhattan distance is usually used in calculating graph-like paths. It is calculated using an absolute sum of difference between its Cartesian coordinates as below:

$$d_1 : (x, y) \rightarrow \|x - y\|_1 = \sum_{i=1}^{n} |x_i - y_i|$$

Chebyshev distance is also known as chessboard distance as it is the distance between two spaces on a chess board that gives the minimum number of moves a king requires to move between them.

$$d_\infty : (x, y) \rightarrow \|x - y\|_\infty = max_i |x_i - y_i|$$

Comparing different metrics in k-NN models, Manhattan distance outperforms Chebyshev distance and the most commonly used Euclidean distance, as shown in Figure 8. This is because Manhattan distance handles data with binary attributes well and our data contains a lot of binary numbers as we use one hot encoding.

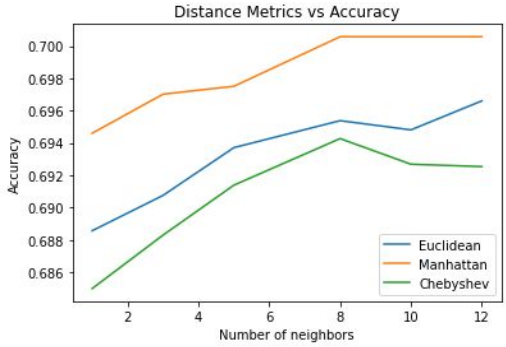

Figure 8. Validation accuracy of k-NN models with different metics.

3.2.2 Weighted k-NN

In regular k-NN, we simply use the majority vote from the nearest neighbors to predict the test data. Since the weights of different neighbors are uniform, this is also called uniform weighted k-NN.

As we increase the number of neighbors k, the generalization increases while the model becomes less stable. When we use a large k (i.e. greater than 8), the accuracy of the uniform k-NN stops increasing. As shown in Figure 9. This is because the local structure can no longer be represented well with large k values.

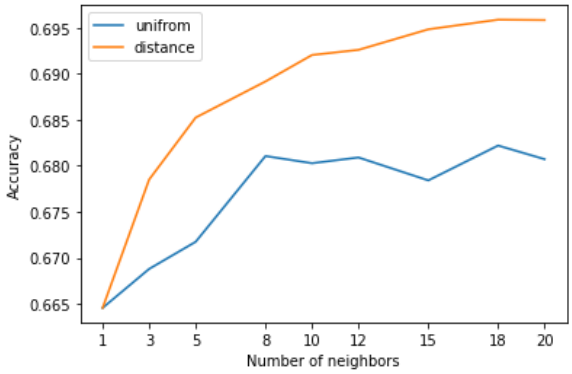

Figure 9. Accuracy of uniform k-NN and distance-weighted k-NN.

Distance-weighted k-NN has better performance than uniform k-NN where each neighbor is weighted by the distance. Closer neighbors represent local features better so they have larger weights. Here we use the inverse of the distance as the weights. The accuracy keeps going up as k increases, even in a large value (i.e. $k = 18$)

3.2.3 Bagging and pasting

We test ensemble learning on k-NN models with bagging methods.

Bagging methods form a class of k-NN models which build several instances of k-NN estimators on random subsets of the original training set. The bagging methods we use here are bagging and pasting. They differ from each other by the way they draw random subsets of samples. Bagging draws subset with replacement while pasting draws without replacement.

The results of single k-NN and k-NN with bagging methods are shown in Table 3.

	Single k-NN	Bagging	Pasting
k=1	0.662	0.673	0.669
k=5	0.673	0.678	0.678
k=10	0.681	0.680	0.680

Table 3.Validation accuracy of single k-NN and k-NN with bagging methods.

For a k-NN model with small k (i.e. $k = 1$), bagging and pasting increase the model performance by reducing variance between estimators. Bagging performs better than pasting.

For a k-NN model with larger k, bagging and pasting don't increase accuracy compared to a single k-NN model. Difference between bagging and pasting is trivial. This is because bagging and pasting benefit unstable learners that are usually sensitive to modified datasets. However, our k-NN models with proper k values are stable and effective therefore don't benefit from bagging methods.

We achieve the highest accuracy of 0.716 on the k-NN model. In the fine-tuned model, we use Manhattan distance and samples are weighted by distance.

3.3. Support Vector Machine

Support Vectors Classifier tries to find the best hyperplane to separate the different classes by maximizing the distance between sample points and the hyperplane. We implement different kernels and tune the hyperparameters to find the best model for severity prediction.

Since SVM models are not scale-invariant, normalizing the input data before applying the model can significantly speed up the training process.

3.3.1 Kernel Chosen and Number of Samples

The kernel functions that might perform well on our dataset are: linear, polynomial (poly), and radial basis function (rbf).

Linear kernel function can be expressed as:

$$< x, x' >$$

With default parameters, its training accuracy vs. the number of training points is shown in Figure 10. The best accuracy is around 0.69.

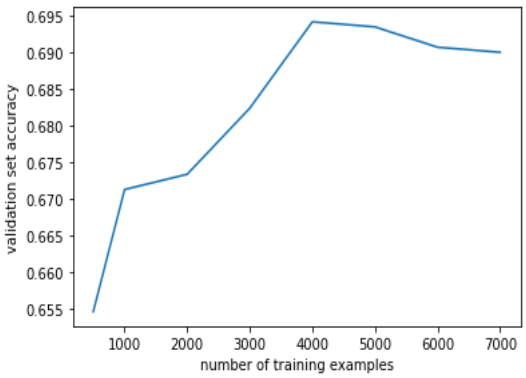

Figure 10. Accuracy of linear SVM vs. the number of training samples.

Polynomial kernel function can be expressed as:

$$(\gamma < x, x' > + r)^d$$

Its training accuracy vs. the number of training points using default parameters is shown in Figure 11. The best accuracy is around 0.72.

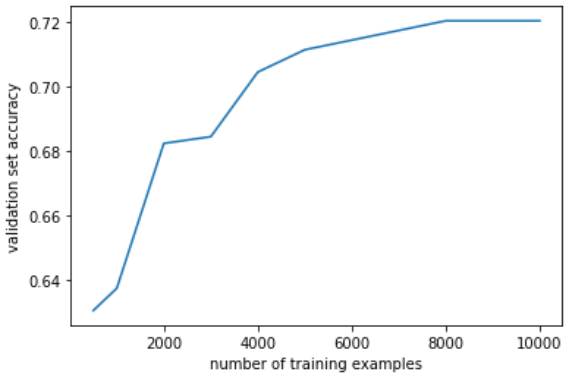

Figure 11. Accuracy of poly SVM vs. the number of training samples.

Radial basis function kernel can be expressed as:

$$exp(-\gamma ||x - x'||^2)$$

Its training accuracy vs. the number of training points is shown in Figure 12. The best accuracy is also around 0.72.

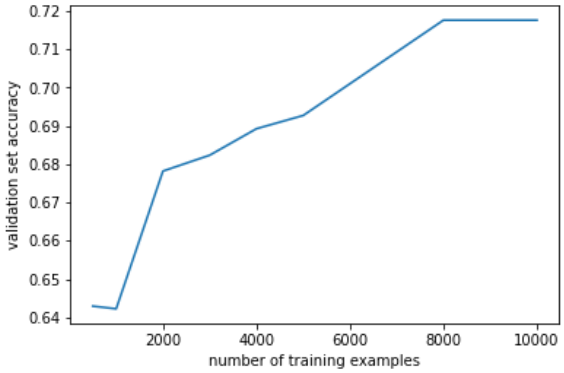

Figure 12. Accuracy of rbf SVM vs. the number of training samples.

Obviously, poly and rbf kernels perform better than the linear model. This is because they use non-linear hyperplanes to classify the sample points. SVM model training is super slow with a large number of data points. From Figure 10,11 and 12, training 8000 samples provides the best tradeoff between accuracy and speed. Therefore, all the following training processes use 8000 training samples.

3.3.2 Hyperparameter Tuning

C is a hyperparameter that controls the strength of regularization. It controls the tradeoff between smooth decision boundary and training accuracy. Figure 13 and 14

show the validation set accuracy vs. different C values for poly model and rbf model, respectively. For both models, the best accuracy occurs when C is around 15.

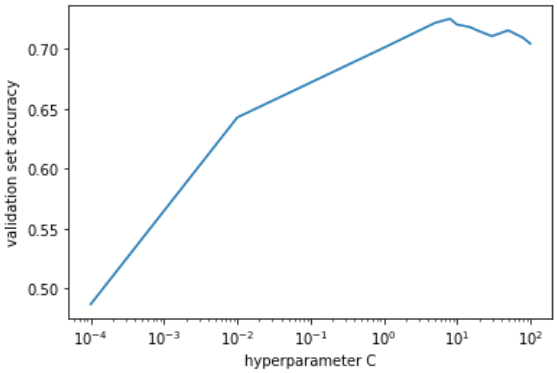

Figure 13. Accuracy of poly SVM vs. C.

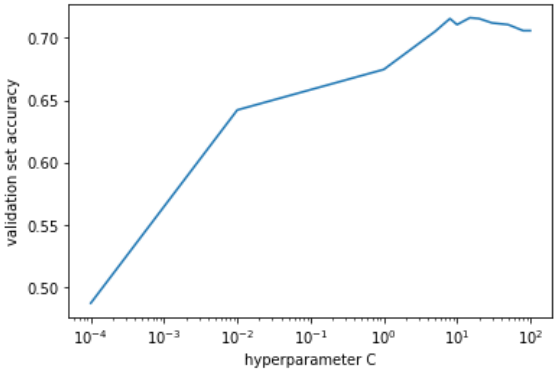

Figure 14. Accuracy of rbf SVM vs. C.

Hyperparameter γ is used for non-linear hyperplane tuning. The higher the γ value is, the harder the model tries to exactly fit the training dataset. Figure 15 and 16 show the validation set accuracy vs. different γ values for poly model and rbf model, respectively. For the poly model, the best accuracy occurs when γ is around 0.02, and the best γ for the rbf model is around 0.04.

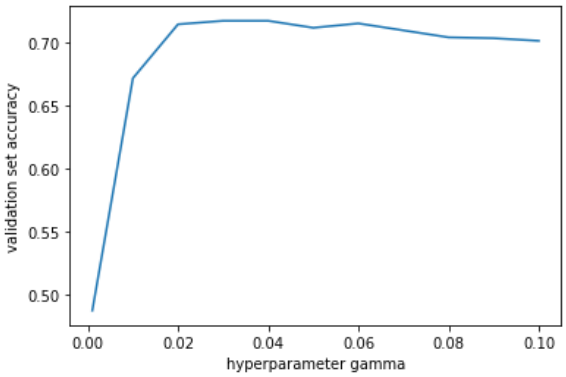

Figure 15. Accuracy of poly SVM vs. γ.

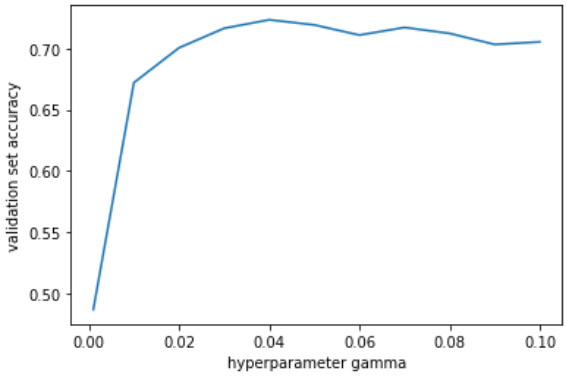

Figure 16. Accuracy of rbf SVM vs. γ.

For the poly model, we also tuned the parameter "degree", which refers to the degree of the polynomial kernel function. The degree is swept from 0 to 6 and the result is shown in Figure 17. The best degree for our application is 2.

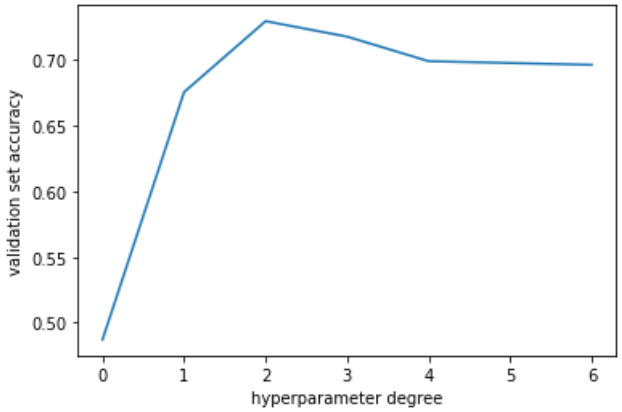

Figure 17. Accuracy of poly SVM vs. degree.

Therefore, with the well-tuned SVM model, we achieved 0.741 test accuracy, when using the poly model with degree = 2, C = 15, and γ = 0.02.

4. Comparison and Discussion

	Decision Tree	K-NN	SVM
Accuracy	0.927	0.716	0.741

Table 4. Accuracy of different models

In our work, we obtain the best accuracy by using decision trees with minimal cost-complexity pruning as shown in Table 4. In this section, we analyze and compare different classifiers in our project and explain why decision trees can obtain the best performance on our data.

Predicting accident severity is a nonlinear multiclass problem. Decision trees are nonlinear algorithms that are designed in a structure intrinsically suited for multiclass situations, while basic support vector machines are mainly designed for linear binary classification problems. The k-NN model is suitable for nonlinear multiclass problems with low feature dimensions. In our data, feature dimensionality is high. Besides, the k-NN algorithm is a clustering algorithm based on neighborhoods. So, we have to use a distance metric and all the features must be numeric. In our data, we have several hot features. These features are treated as numeric values and can negatively influence the final performance.

5. Conclusions and Future Work

In this paper, we used 500,000 data points to predict the severity of car accidents in California. We drop meaningless features from 49 features in raw data. We simplify text data into key words and use one-hot encoding to convert them into numerical data. Finally, we obtain data with 42 features.

Decision tree, k-NN and SVM models are used for prediction. With decision trees, we tune minimum leaf size and split threshold to decide when the tree stops dividing. To balance the influence of different classes, we use balanced weights. We also prune unnecessary branches of a complete tree to avoid overfitting. For k-NN, we compare models with different distance metrics. Distance-weighted k-NN has better performance than uniform k-NN. In a distance-weighted knn, the closer the neighbour, the more important it is. Thus, we can preserve generalization when we increase the number of neighbors. For SVM, We tried linear, poly, and rbf kernels. Poly and rbf have higher accuracy than the linear model. We tuned the penalty term of the error and γ for poly and rbf kernels to get better trade-off between underfitting and overfitting.

Among all the models, the decision tree with minimal cost-complexity pruning achieves the highest accuracy at 0.927.

Reference

[1] World Health Organization. 2015. *Global status report on road safety 2015*. World Health Organization.

[2] US Accidents (3.0 million records) A Countrywide Traffic Accident Dataset (2016 - 2019) https://www.kaggle.com/sobhanmoosavi/us-accidents

[3] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, pp.2825-2830, 2011.

[4] Altman, Naomi S. 1992. *"An introduction to kernel and nearest-neighbor nonparametric regression"*. The American Statistician. 46 (3): 175–185.