# Predicting the severity of US car accidents from (2016 - 2021)

## *1. Introduction*

Road accidents cost the United States hundreds of billions of dollars in economic and societal damage every year. Furthermore, a small number of large incidents account for a significant share of the losses. However, preventing traffic accidents, especially significant ones, is never an easy undertaking. The proactive technique, which focuses on preventing potentially dangerous road conditions from occurring in the first place, is one of the two main approaches to dealing with traffic safety concerns. The effectiveness of this method hinges on the ability to foresee accidents and their severity. Many factors affect the severity of the accident; in the data set, the variables that we consider that affect the severity are:

1. ID- The unique identifier of the accident; 2. Severity; 3. Start Time; 4. End Time; 5. Start Latitude; 6. Start Longitude; 7. End Latitude; 8. End Longitude; 9. Distance in Miles; 10. Description of Accident; 11. The number of the Street; 12. Street Name; 13. Side of the road; 14. City; 15. County; 16. State; 17. Zip Code; 18. Country; 19. Time Zone; 20. Airport Code which denotes nearest airport-based weather station; 21. Weather Timestamp; 22. Temperature; 23. Wind Chill; 24. Humidity; 25. Pressure; 26. Visibility; 27. Wind Direction; 28. Wind Speed; 29. Precipitation; 30. Weather Conditions; 31. Presence of amenity in nearby location; 32. Bump near location; 33. Crossing; 34. Presence of giving way location nearby; 35. Presence of Junction nearby; 36. No Exit sign nearby; 37. Presence of Railway nearby; 38. Presence of Roundabout nearby; 39. Presence of station nearby; 40. Presence of Stop sign nearby; 41. Presence of Traffic Claiming sign nearby; 42. Presence of Traffic Signal sign nearby; 43. Presence of Turning Loop nearby; 44. Period Sunrise/Sunset; 45. Time of Civil Twilight; 46. Time of Nautical twilight; 47. Time of Astronomical twilight.

### *a) The reasoning for the choice of Project*

The first goal is to find out the factors that affect the severity. The second goal is to create a model that can reliably forecast the severity of an accident. To be more explicit, this model is expected to predict the chance of a severe accident to a particular accident without any comprehensive information about the accident, such as driver qualities or vehicle type. For example, the catastrophe might have occurred recently, and details are still unavailable, or it could be one anticipated by other models. As a result, this model may predict serious accidents in real-time using the advanced real-time traffic accident prediction system built by the inventors of the same data-set utilized in this study.

Furthermore, a small number of significant accidents account for a considerable portion of the losses. Nonetheless, reducing road accidents, preeminent ones, is always a difficult task. One of the two primary ways of dealing with traffic safety issues is the proactive strategy, which focuses on preventing potentially dangerous road situations in the first place. Accident prediction and severity prediction are essential for the successful execution of this strategy.

### *b) The explanation for projects Application.*

The data has a wide range of applications; it can predict patterns in accidents, study accident locations and direct them to hospital locations. It can be used to make important policy decisions to reduce the number of accident severity. This data can also compare accidents caused by human driving patterns and accidents caused by self-driving cars.

### *c) Source of data set.*

- https://www.kaggle.com/sobhanmoosavi/us-accidents\
- Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. "A Countrywide Traffic Accident Dataset.", 2019.
- Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. "Accident Risk Prediction based on Heterogeneous Sparse

Data: New Dataset and Insights." In Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

## 2. *Data Cleaning:*

### *a) Categorical variables in the data set and the reasoning behind generating dummy variables:*

Categorical variables such as Weather conditions and Wind direction were given dummy values to handle the data better. For example, there were 11 unique weather conditions like rain, cloudy, hail, snow, etc. Weather conditions play an essential role in the severity of accidents, so this data could not be excluded; therefore, dummy variables were assigned. In addition, there was ten unique wind direction for which dummies were given as well.

### *b) Ways in which missing values were handled*

The variables such as street number, wind chill, and precipitation have more than 20% of their value missing; therefore, we will drop these variables. We are also creating a new column for the difference between end and start times for a time duration. Again, the outliers and the negative values for time duration were removed.

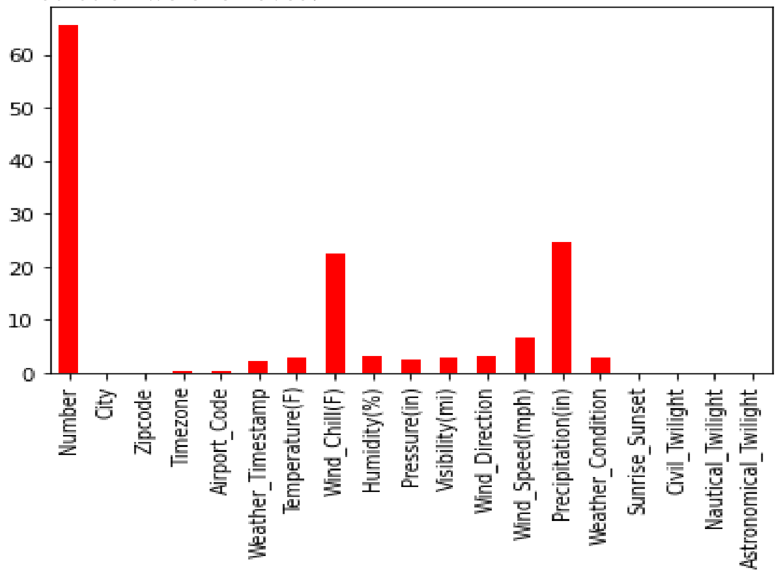


Figure 1: The variables with percentage missing values.

### *c) Bin generation for the data and Feature Encoding*

Information such as sunrise/sunset, civil twilight, Nautical Twilight, Astronomical Twilight could only have one of two outcomes. Therefore, these variables were given binary values of either 0 or 1. We can encode the categorical features using the method get dummies() which converts the features with the one-hot encoding. This creates a binary column for each category and returns a sparse matrix or dense array (depending on the sparse parameter).

### *d) Standardization and normalization that was carried out for the variables*

Values with higher numbers could significantly influence the data. They could cause biases in our analysis, so we had to normalize and standardize this data to assume values between 0 and 1. For this purpose, we will use Min_Max_Scaler for temperature, humidity, pressure, Pressure, Visibility, Wind speed, Starting Longitude, and Ending Longitude.

### *e) Randomly selecting data from the data-set*

We used the random function to select the data from our data set randomly. We initially had one million rows and forty-seven columns. However, solving that was not possible because of a lack of computational power. Hence, we used a random function and shrank the data set to make it easier for work.

## 3. *Preliminary Analysis:*

### a) *Matrix scatter plot for the variables*

A Scatter plot matrix is a matrix (or grid**)** of scatter plots where each scatter plot in the grid is created between different combinations of variables. In other words, the scatter plot matrix represents **the** bi-variate or pairwise relationship between different combinations of variables while laying them in grid form. For example, from the plot below, our data is highly non-linear or skewed.

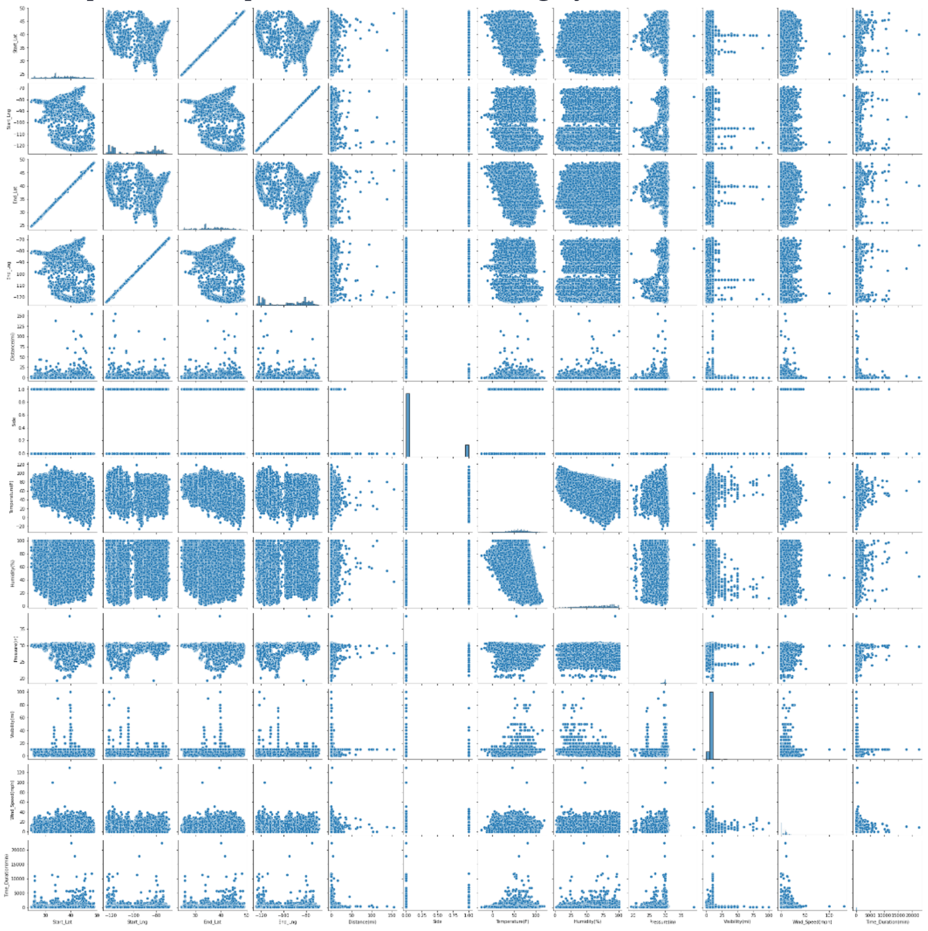


Figure 2: Scatter plot between variables

### b) *Response -predictors and predictor-predictor pairwise correlation*

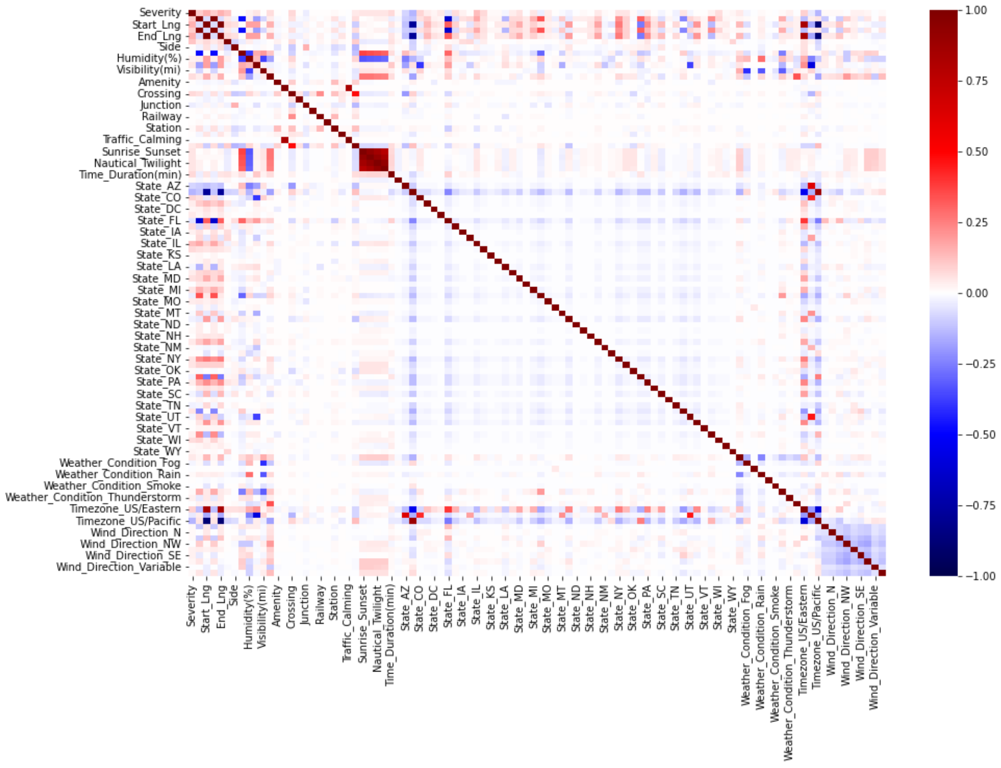


Figure 3: Correlation Matrix for response-predictor and predictor-predictor.

From the correlation matrix, we can observe some multicollinearity. There is a positive correlation between the Starting and Ending Longitudes and Latitudes, as the value is higher than 0.7.

- <u>Predictor-predictor</u>

High positive correlations between two pairs of predictor variables can be observed. **Start_Lat and End_Lat** are highly correlated. **Start_Lng and End_Lng** also have collinearity between each other, as the value is higher than 0.7. There is a negative relation between **Temperature** and **Humidity** (-0.4). This is not a strong relation but still worth mentioning.

### *c) Complications that could occur with the data:*

The model will be complex because the data-set is huge, and understanding them would require more advanced Machine Learning techniques or Neural Networks. Also, due to the limited computational power, we needed to reduce the data-set; hence this adds some bias to our results. Furthermore, we can see some multicollinearity in the data that can hinder the performance as well. This was seen from the correlation matrix.

## *4. Modeling Techniques:*

In the Performance Evaluation section, all the modeling methodologies utilized in this Project are presented, together with their parameters and hyperparameters, and then assessed and interpreted based on their performance using relevant metrics. Additionally, the performance of these fitted models were also evaluated. Furthermore, training and testing mistakes were compared. Finally, hyperparameter optimization was carried out with the help of different models, and there is a trade-off between bias and variance.

### <u>**Logistic Regression Model:**</u>

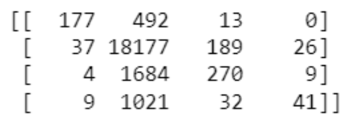
At times, we need to classify a dependent variable that has more than two classes. For this purpose, the binary logistic regression model offers multinomial extensions. As a result, multinomial logistic regression analysis has many aliases: polytomous LR, multiclass LR, softmax regression, multinomial logit, and others.

### *a) Explain all the Parameters and Hyper-Parameters in detail:*

- Solver = ‘newton_cg’; ’sag’; ‘saga’; ‘lbfgs’ are used in multiclass problems to handle multinomial losses.
- Penalty = ‘none’; ‘j’ is used to add a penalty. There was no improvement in the result when the penalty was added.
- fit_intercept = ‘True’; ’False’ specifies if a constant (bias or intercept) should be added to a function
- class_weight = None means that no weight was added and all classes are supposed to have weight one.
- random_state = 0/1, Used when we need to shuffle data for ‘sag’; ‘saga.’
- n_jobs= -1, Number of CPU cores used when parallelizing over classes when multi_class = ‘ovr.’
- multi_class = ‘ovr’ means the binary problem is a fit for each label. For ‘multinomial,’ the loss minimized is the multinomial loss fit across the entire probability distribution.
- max_iter=10000 Maximum number of iterations it takes for the solver to converge.

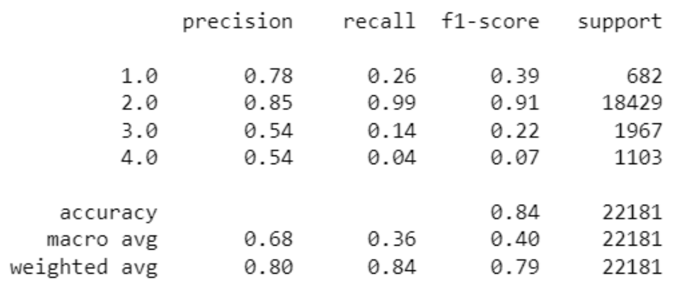
### *b) Evaluating the performance of the models:*

Weighted Average Precision Accuracy was chosen as the ‘Y’ labels are not evenly distributed, the score for which is 0.80. Therefore, the score is good for Logistic Regression. The classification report and the confusion matrix for Logistic Regression are provided below.



```
[[  177   492    13     0]
 [   37 18177   189    26]
 [    4  1684   270     9]
 [    9  1021    32    41]]
```

Table 1: Confusion matrix of Logistic Regression



```
              precision    recall  f1-score   support

         1.0       0.78      0.26      0.39       682
         2.0       0.85      0.99      0.91     18429
         3.0       0.54      0.14      0.22      1967
         4.0       0.54      0.04      0.07      1103

    accuracy                           0.84     22181
   macro avg       0.68      0.36      0.40     22181
weighted avg       0.80      0.84      0.79     22181
```

Table 2: Classification Report for Logistic Regression

***c) Compare the performance of fitted models (i.e., training error vs. testing error):***

- The training prediction accuracy for Logistic Regression is 0.843
- The testing prediction accuracy for Logistic Regression is 0.841
- train mean squared error= 0.307169799978961
- test mean squared error= 0.3091384518281412
- Precision Average Score= 0.80

***d) Perform bias-variance trade-off for hyperparameter optimization of the models:***

The error remains constant for all the different solvers used like ‘newton_cg,’ ‘lbfgs,’ ‘sag,’ ‘saga.’ Therefore, optimizing the penalty score also does not affect both the training and testing model.

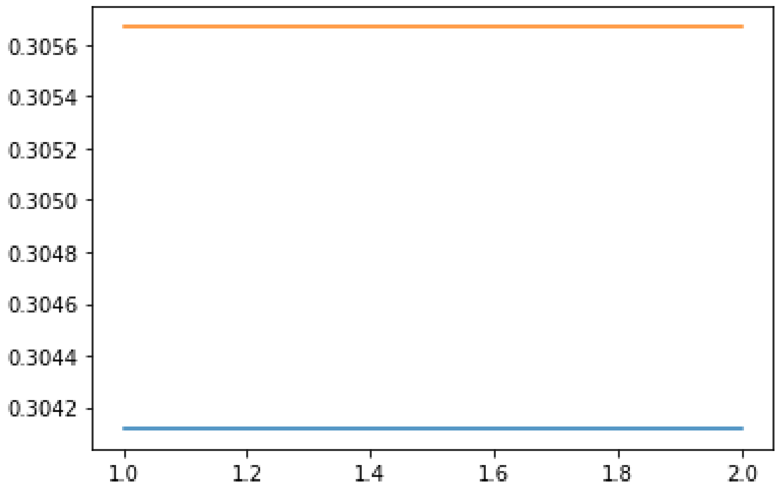


Figure 4: MSE for Logistic Regression training and testing with different penalty terms

<u>**Linear Discriminant Analysis:**</u>

Linear Discriminant Analysis is a linear decision boundary classifier created by fitting class conditional densities to data using Bayes' rule. The model gives each class a Gaussian density, which assumes that all classes have the same covariance matrix. Using the transform approach, the fitted model may be utilized to minimize the dimensionality of the input by projecting it to the most discriminative directions.

***a) Explain all the parameters and hyperparameters:***

- Solver=’svd’ Singular Value decomposition (default) does not compute the covariance matrix. Hence it is the solver used when data has a large number of features.
- Shrinkage=’None’ has no shrinkage. Also, shrinkage only works with ‘lsqr’ and ‘Eigen solvers.’
- Priors=’None’ By default, the class proportions are inferred from the training data.
- Store_covariance=’False’ If True, explicitly compute the weighted within-class covariance matrix when the solver is ‘svd.’ The matrix is continuously calculated and stored for the other solvers.

***b) Evaluating the performance of the models:***

Three grid solvers were used; ‘svd,’ ‘lsqr,’’eigen,’ and the shrinkage was 0.0, the accuracy of the training and testing of the model 0.826 and 0.823, respectively. Precision data shows that LDA considerably reduces the type 1 error (actual positive observed out of the total positive results). The Precision Accuracy has the lowest Precision Accuracy Scores. The classification report and the confusion matrix for LDA are provided below.

```
[[   204   457    20     1]
 [   343 17473   407   206]
 [    21  1428   456    62]
 [    31   876    61   135]]
```

Table 3: Confusion matrix of LDA

```
              precision    recall  f1-score   support

         1.0      0.341     0.299     0.319       682
         2.0      0.864     0.948     0.904     18429
         3.0      0.483     0.232     0.313      1967
         4.0      0.334     0.122     0.179      1103

    accuracy                          0.824     22181
   macro avg      0.505     0.400     0.429     22181
weighted avg      0.787     0.824     0.797     22181
```

Table 4: Classification Report for LDA

***c) Compare the performance of fitted models (i.e., training error vs. testing error):***

- The training prediction accuracy for LDA is 0.826
- The testing prediction accuracy for LDA is 0.823
- train mean squared error for LDA= 0.337
- test mean squared error for LDA= 0.340
- Precision Average Score= 0.787

***d) Perform bias-variance trade-off for hyperparameter optimization of the models:***

After using grid search solver’s svd, lsqr, and eigen, it was found that svd (single value decomposition). Gave the best results. The mean accuracy for it was 0.825. Also, shrinkage offers no improvement; therefore, it was set to 0.0.

**<u>K-Nearest Neighbor:</u>**

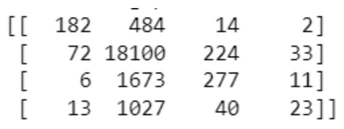
The idea behind nearest neighbor approaches is to locate a certain number of training samples closest to the new point and use them to forecast the label. The number of samples can be a user-defined constant (k-nearest neighbor learning). On the other hand, it can vary depending on the density of points in the immediate area (radius-based neighbor learning). In general, any metric measure can be used, although the conventional Euclidean distance is the most used.

***a) Explain all the parameters and hyperparameters:***

- n_neighbors=10;24. As the name suggests, the number of neighbors is 10 and 24 for kneighbors queries.
- weights=’uniform’. This option is used to give neighbors a weighted average. This option is set to 'distance' if closer neighbors should have a greater effect than others. Weights are assigned to all of the neighbors in the same way.
- p=2 Power parameter for the Minkowski metric. When p=1, this is equivalent to using manhattan_distance (l1), and euclidean_distance(l2) for p=2. For arbitrary p, minkowski_distance (l_p) is used.
- Metric=’minkowski’ The distance metric to use for the tree. The default metric is minkowski, and p=2 is equivalent to the standard Euclidean metric. See the documentation of Distance Metric for a list of available metrics. If the metric is “precomputed,” X is assumed to be a distance matrix and must be square during the fit. X may be a sparse graph, in which case only “nonzero” elements may be considered neighbors.

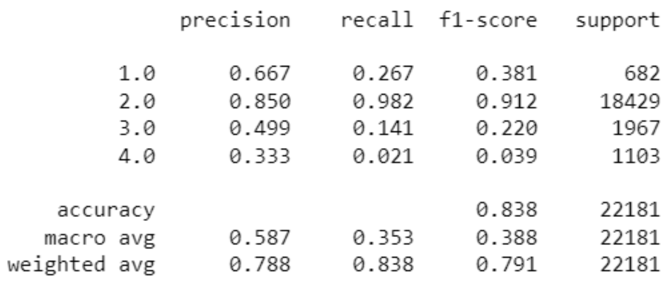
***b) Evaluating the performance of the models:***

KNN has a little higher mean accuracy on training and testing than the other models, with 0.851 and 0.837, respectively, but we know this isn't the only factor we're considering when selecting our top model. So, along with the training and testing mistakes, an average precision score for all the severity is discovered, which is among the most outstanding scores among all models. Because the difference between the testing and training errors is modest, this model may have low bias and variance. The classification report and the confusion matrix for KNN are provided below.



```
[[   182   484    14     2]
 [    72 18100   224    33]
 [     6  1673   277    11]
 [    13  1027    40    23]]
```

Table 5: Confusion matrix of KNN



```
              precision    recall  f1-score   support

         1.0      0.667     0.267     0.381       682
         2.0      0.850     0.982     0.912     18429
         3.0      0.499     0.141     0.220      1967
         4.0      0.333     0.021     0.039      1103

    accuracy                          0.838     22181
   macro avg      0.587     0.353     0.388     22181
weighted avg      0.788     0.838     0.791     22181
```

Table 6: Classification Report for KNN

***c) Compare the performance of fitted models:***

- The training prediction accuracy for KNN is 0.851
- The testing prediction accuracy for KNN is 0.837
- train mean squared error= 0.303
- test mean squared error= 0.304
- Precision Average Score= 0.788

***d) Perform bias-variance trade-off for hyperparameter optimization of the models:***

The optimal number of neighbors for the KNN model was chosen to be 31 using GridSearchCV and cross_val_score. The 10-fold cross-validation score was 0.845 (Training) and 0.840 (Testing), n = 31 was taken as a value of k for KNN from the graph shown below based on which the KNN model was fit.

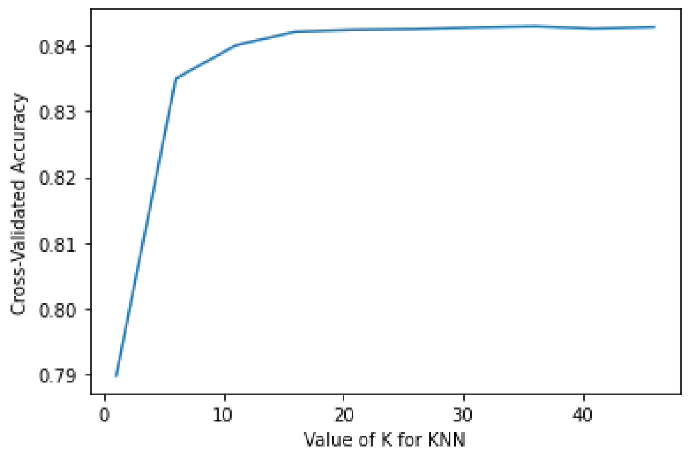


Figure 5: Value of K for KNN vs. Cross Validated Accuracy

**<u>Decision Tree Classifiers (including Random Forest):</u>**

The Decision Tree Classifier is a supervised machine learning technique for classification problems that learns basic decision rules from the training data to predict target variables. This strategy is straightforward to comprehend and interpret, as well as visualize. However, this approach is insecure since slight changes in the data-set might result in an entirely new tree being produced. Furthermore, because optimum judgments are taken locally at each node, this strategy is called "greedy," It does not necessarily provide the globally optimal decision tree. The Random Forest Classifier is an ensemble approach in which many decision tree classifiers are fitted to subsets of the data-set. It uses all subgroups to forecast the target variable and then averages the final prediction results.

***a) Explain all the parameters and hyperparameters:***

- Criterion: 'gini' The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
- Splitter: 'best' The strategy used to choose the split at each node. Supported methods are "best" to choose the best split and "random" to choose the best random split.
- max_depth: 32 This determines the decision tree's maximum depth or expansion. Using RandomizedSearchCV between 0 and 100 and 'None,' the max depth was set at 32. If 'None' is used, the tree will grow until all leaves have fewer samples than the min samples split.
- min_samples_split: '26'; This is the minimum number of samples in the internal node after which it can be split. The min_samples_split was chosen as 26 using RandomizedSearchCV between the range

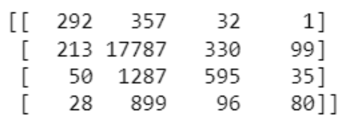
0 and 100.

- min_samples_leaf: ‘16’ The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least one training sample in each left and right branch. This may have the effect of smoothing the model, especially in regression.
- min_weight_fraction_leaf: ‘0.0’ The minimum weighted fraction of the total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- max_features: ‘None’ The maximum number of features to consider when looking for the best split is None.
- max_leaf_nodes: ‘None’ There is an unlimited number of leaf nodes.
- class_weight: ‘None’ All classes are supposed to have weight one.
- ccp_alpha: ‘0.0’ Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than 0 will be chosen.

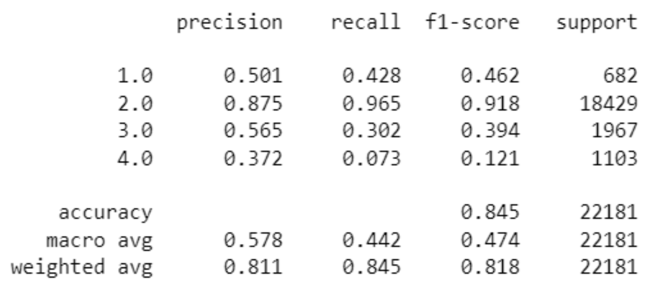
***b) Evaluating the performance of the models:***

For Decision Tree Classifier

The precision accuracy of the Decision Tree Classifier for training and testing is 0.999 and 0.833, respectively. Which means the data has high variance and low bias. But the precision score is 0.811. The confusion matrix and Classification Report for the Decision Tree Classifier are given below.



```
[[  292   357    32     1]
 [  213 17787   330    99]
 [   50  1287   595    35]
 [   28   899    96    80]]
```



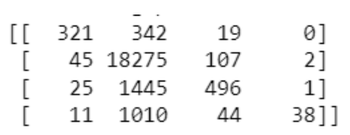
```
              precision    recall  f1-score   support

         1.0      0.501     0.428     0.462       682
         2.0      0.875     0.965     0.918     18429
         3.0      0.565     0.302     0.394      1967
         4.0      0.372     0.073     0.121      1103

    accuracy                          0.845     22181
   macro avg      0.578     0.442     0.474     22181
weighted avg      0.811     0.845     0.818     22181
```

Table 7: Confusion matrix of Decision Tree

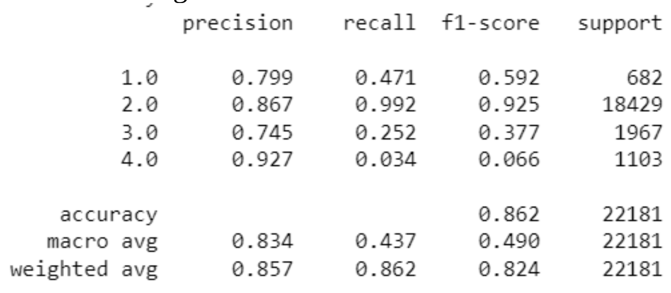
Table 8: Classification Report for Decision Tree

For Random Forest Classifier

The random classifier is the best performing model and has the best precision accuracy score of 0.857. The MSE also has the lowest value for training as well as testing data-set. The confusion matrix and Classification Report for the Random Forest Classifier are given below.



```
[[  321   342    19     0]
 [   45 18275   107     2]
 [   25  1445   496     1]
 [   11  1010    44    38]]
```



```
              precision    recall  f1-score   support

         1.0      0.799     0.471     0.592       682
         2.0      0.867     0.992     0.925     18429
         3.0      0.745     0.252     0.377      1967
         4.0      0.927     0.034     0.066      1103

    accuracy                          0.862     22181
   macro avg      0.834     0.437     0.490     22181
weighted avg      0.857     0.862     0.824     22181
```

Table 9: Confusion matrix of Random Forest

Table 10: Classification Report for Random Forest

***c) Compare the performance of fitted models (i.e., training error vs. testing error):***

For Decision Tree Classifier

- The training prediction accuracy for the Decision Tree is 0.999
- The testing prediction accuracy for the Decision Tree is 0.833
- train mean squared error= 0.287
- test mean squared error= 0.311
- Precision Average Score= 0.811

For Random Forest Classifier

- The training prediction accuracy for Random Forest is 0.8761702958988924
- The testing prediction accuracy for Random Forest is 0.862449844461476
- train mean squared error= 0.26400973806410893
- test mean squared error= 0.2843424552544971

- Precision Average Score= 0.857

***d) Perform bias-variance trade-off for hyperparameter optimization of the models:***
The ideal parameters for the Decision Trees Classifier and the Random Forest Classifier were found usin g a RandomizedSearchCV. Because decision trees tend to overfit the training data, they have a low bias a nd, as a result, a large variance. Random Forests attempt to reduce variance by generating many trees and averaging together many weakly connected trees, lowering the correlation between them and reducing ov erfitting. The bias-variance trade-off was discovered by trial and error using hyperparameters like max fe atures, min samples leaf, and min samples split. Increased min samples leaf reduces variance by preventi ng the Decision Tree from creating branches for every case. The model would not overfit the data if the minimum samples in the leaf node were higher, which was determined to be 11 for a random forest class ifier. This is also true for min samples split.

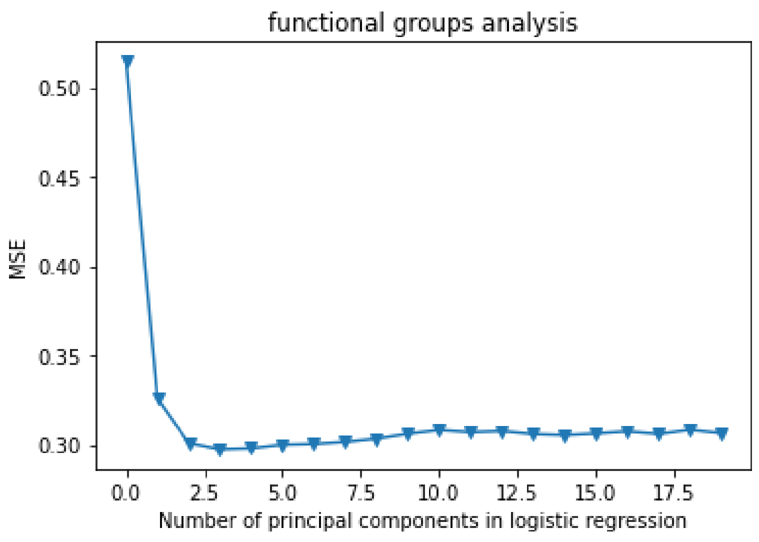
| | **Decision Tree Classifier** | **Random Forest Classifier** |
|---|---|---|
| Min sample split | 36 | 9 |
| Min sample leaf | 14 | 11 |
| Max features | Auto | Auto |
| Max depth | 78 | 79 |
| Criterion | gini | Bootstrap=False |
| Model Score (Training) | 0.852 | 0.863 |

Table 11: Values of Parameters and hyperparameters for Decision Tree and Random Forest Classifier

## *5. Model Selection:*

***a) Applying Ridge shrinkage methodology and PCA dimension reduction technique on the data-set. Results after performing cross-validation to find the best model and performing a bias-variance trade-off for model flexibility***

Ridge and Lasso's regression are simple techniques to reduce model complexity and prevent over-fitting, resulting from simple linear regression. The value of $R^2$ for ridge shrinkage is 0.12 for both testing and training data. Also, the MSE of training and testing are 0.257 and 0.258. The values of MSE for Lasso shrinkage are 0.257 and 0.259 for testing and training data, respectively. Both regression models show poor results, which would mean that the data-set is highly non-linear.



Principal component analysis involves reducing the dimension of the original dataset into ‘n’ number of principal components which helps in reducing the multi-collinearity problem and thereby increasing the model performance and prediction capability. The above Figure shows that the best value for the number of principal components was chosen as 3 as it has the lowest MSE.

### *b) Finding the optimal hyperparameter for the selected model using cross-validation*

In the same manner that Random Forest was used to discover the ideal hyperparameters of the data-set with reduced features. RandomizedSearchCV was used to find the optimal hyperparameters of the data-set with decreased features. On the smaller data-set, this model with improved hyperparameters was fitted and assessed the same way as the previous case.

### *c) Best overall model*

Out of the five fitted models, the Random Forest Classifier was picked as the best. As previously stated, precision and the difference between training and testing errors were chosen as relevant metrics to evaluate all fitted models. Moreover, while Logistic Regression and K-Nearest Neighbors had higher accuracy ratings among the fitted models, the difference in errors was greater than with Random Forest. Because the training and testing accuracy differ so little, and the training accuracy is 0.876, which is considered an excellent match, this model has a low bias compared to the others. Furthermore, the testing accuracy is 0.862, which is extremely similar to the training accuracy, indicating that the model is doing well in explaining variation in the testing data-set, indicating that the model has low variance compared to other models. Aside from that, the average accuracy score is also excellent.

### *d) Comparing the model results with the results from the correlation matrix*

| **Full Dataset** | **Reduced Dataset using PCA** |
|---|---|
| Mean Accuracy (Training): 0.876 | Mean Accuracy (Training): 0.888 |
| Mean Accuracy (Testing): 0.862 | Mean Accuracy (Testing): 0.842 |
| Precision Avg. Score: 0.857 | Precision Avg. Score: 0.809 |

After fitting the Random Forest with the optimal hyperparameters for the reduced dataset, we can observe that there is no improvement in the performance of the model in the bias-variance trade-off and precision average score.

As the difference between the training and testing accuracies is higher in the reduced dataset, it means that the model has not improved in performance in predicting the testing data. This model has high bias and variance when compared to the old model. This means that most of the features that are lost are not transformed or projected on the fewer principal components. The precision average has also reduced which suggests that the model didn't do well in predicting more True positive values accurately which ultimately increased the False Positive values.

## *6. Conclusion:*

The severity of accident prediction was carried out using five distinct machine learning models based on various functional categories. Using the correlation matrix and heat map, a minor multicollinearity issue was discovered between the predictors, but for the most part the predictors were independent. In terms of performance, all five machine learning models were comparable. Still, Random Forest outperformed the others in all performance assessment criteria such as precision score, training and testing errors, and accuracies.

The bias-variance trade-off was used to improve the Random Forest hyperparameters, then fitted to the data-set to obtain the best model performance. Compared to other models, this model had a low bias and variance since the difference between the training and testing errors and accuracies were quite small. Also, it was seen that the precision score increased up to 0.86 which means that the model predicted more True Positive values accurately, ultimately lowering the False Positive values. Future work includes adding more fuels to this existing dataset to improve the range of fuels covered.

Further, PCA was used as the dimensionality reduction technique to reduce the multi-collinearity problem and increase the performance of the model. 98 features were reduced to 30 principal components which were used to explain the variation in the dataset. The reduction is not so significant as the number of PCA

dimension is significantly large, it is not very low hence we can say that there is only little effect of multi-collinearity.
As the difference between the training and testing accuracies is higher in the reduced dataset, it means that the model has not improved in performance in predicting the testing data. This model has high bias and variance when compared to the old model. This means that most of the features that are lost are not transformed or projected on the fewer principal components. The precision average has also reduced to 0.809 which suggests that the model didn’t do well in predicting more True positive values accurately which ultimately increased the False Positive values.