US ACCIDENTS

Group:10 Project Report IE 594 DS 1

1.INTRODUCTION.

Every year, road accidents cause economic and societal damage to the tune of hundreds of billions of dollars in the United States. Additionally, a sizable portion of the losses are attributable to a small number of major accidents. However, avoiding traffic accidents, particularly serious ones, is never simple. One of the two main methods for addressing traffic safety issues is the proactive methodology, which focuses on preventing potentially hazardous road situations from happening in the first place. The ability to predict accidents and their severity is essential for this strategy to be effective. The following variables are those that we consider in the data set that have an impact on accident severity:

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 Choice of Project:

The initial objective is to identify the variables that influence severity. The second objective is to develop a model that can accurately predict how serious an accident will be. To be more precise, this model is designed to forecast the likelihood of a serious accident occurring in a certain accident without having any detailed accident-related information, such as driver characteristics or vehicle type. For instance, the disaster might have happened lately and no information is yet available, or it might be one that was predicted by other models. As a result, using the sophisticated real-time traffic accident prediction system developed by the creators of the same dataset used in this study, this model may be able to anticipate serious accidents in real-time.

b) The explanation for projects Application:

The data can be used for a variety of purposes, including accident pattern prediction, accident location analysis, and hospital location guidance. Important policy choices can be made using it to lessen the severity of accidents. This information can be used to compare accidents brought on by human driving habits versus accidents brought on by autonomous vehicles.

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:

To help manage the data, dummy values were assigned to categorical variables including Weather conditions, wind direction, and State. There were 11 different types of weather, including snow, hail, rain, and cloudy weather. This data could not be eliminated since weather plays a crucial effect in the severity of accidents; as a result, dummy variables were assigned. The State must also be known because the data set includes 49 states, each of which was encoded one-hot. Dummies were also assigned to each of the ten distinct wind directions.

b) The methods used to deal with missing values:

We will omit variables like street number, wind chill, and precipitation because they are lacking more than 20% of their value. We are also adding a new column for the time interval's difference between start and end times. Once more, the outliers and the negative time duration values were eliminated.

c) Bin generation for the data and Feature Encoding:

There are only two possible outcomes for information on sunrise/sunset, civil twilight, nautical twilight, and astronomical twilight. Therefore, binary values of 0 or 1 were assigned to these variables. The method get dummies () converts the features with one-hot encoding, allowing us to encode the category characteristics. For each category, a binary column is created, and a sparse matrix or dense array is produced (depending on the sparse parameter).

d) The variables standardization and normalization procedures:

Higher values may have a substantial impact on the data. We have to normalize and standardize this data to assume values between 0 and 1, as they might introduce biases into our analysis. We will utilize Min Max Scaler for starting and ending longitudes as well as temperature, humidity, pressure, visibility, and wind speed.

e) Randomly selecting data from the data-set:

We randomly select the data from our data set using the random function. In the beginning, we had 47 columns and 1 million rows. But due to a shortage of computer capacity, it was unable to solve that. To make things simpler for work, we therefore employed a random function and shrunk the data set.

3. PRELIMINARY ANALYSIS.

a) Matrix scatter plot for the variables:

A scatter plot matrix is a matrix (or grid) of scatter plots, each of which is produced using a distinct set of variables. In other words, the scatter plot matrix displays, in grid form, the bi-variate or pairwise relationship between various combinations of variables. For instance, the plot below shows how skewed or severely non-linear our data.

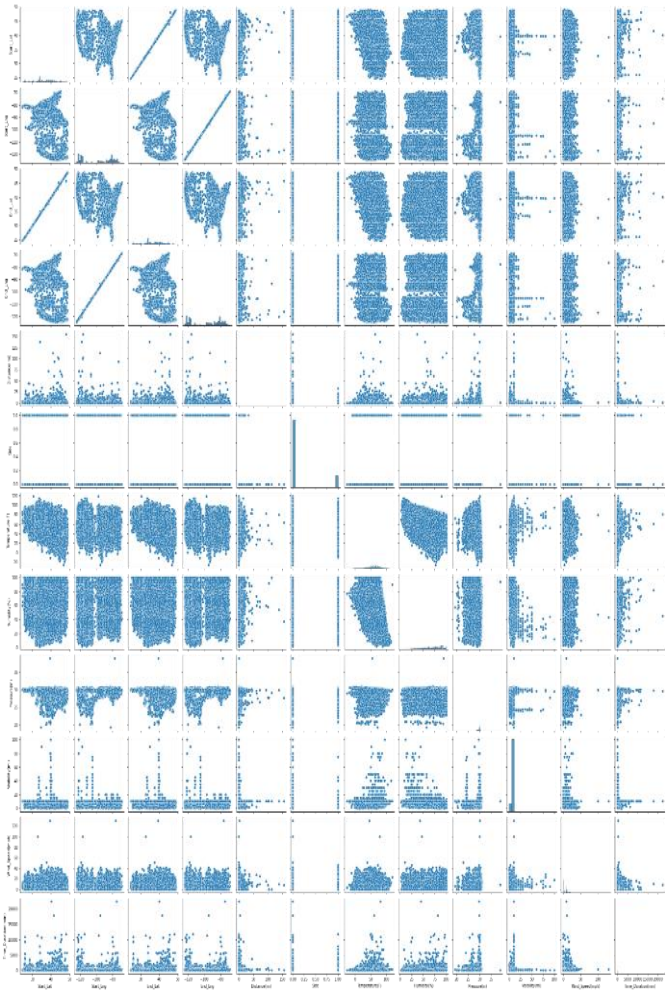

Figure 1: Scatter plot between variables

b) Response -predictors and predictor-predictor pairwise correlation:

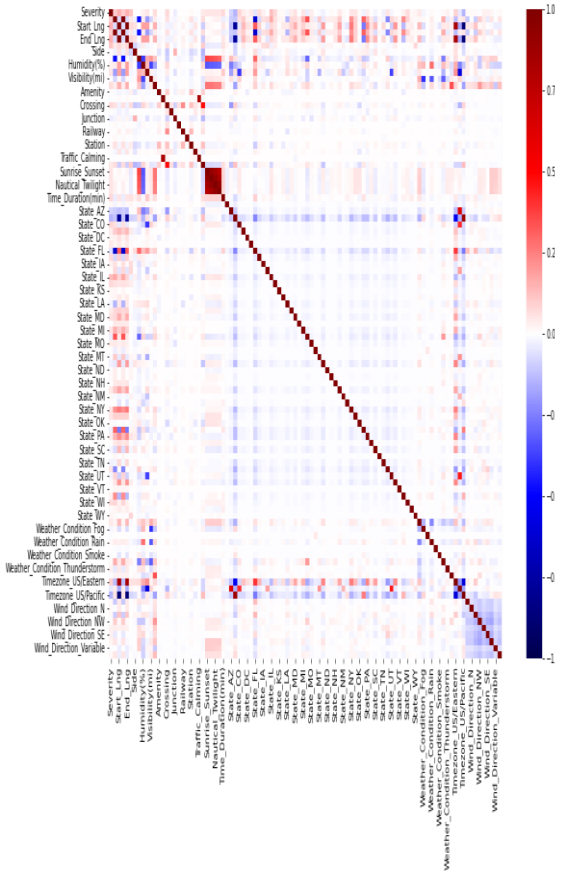

Figure 2: Correlation Matrix for response-predictor and predictor-predictor.

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 0.75.

- **Predictor-predictor**

It is possible to see strong positive correlations between two sets of predictor variables. The start and end times have a strong correlation. Given that the value is more than 0.7, Start Lng and End Lng also exhibit collinearity. Humidity and temperature have a negative relationship (-0.4). Even though there isn't much of a connection, it is still important

- **Response-predictor**

Since the majority of the variables have modest correlations with the answer, the data do not exhibit multicollinearity.

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 of the modelling approaches used in this Project are described, together with their parameters and hyperparameters, and then rated and interpreted based on their performance using relevant metrics. The effectiveness of these fitted models was also assessed. Additionally, errors made during training and testing were compared.Finally,hyperparameter optimization was performed using several models, and there is a trade-off between bias and variance.

1) Logistic Regression Model:

Using multinomial logistic regression, one may predict whether a target variable has more than two classes. It is a variation on logistic regression that uses the SoftMax function in place of the cross entropy loss function's sigmoid function. All values are condensed by the SoftMax function to the range [0,1], and the element sum is 1.

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:

Average Weighted Precision Accuracy was chosen because the 'Y' labels are not distributed equally, with a score of 0.80. As a result, the result for Logistic Regression is favourable. Below are the classification report and confusion matrix for Logistic Regression.

```
[[  103   594     8     0]
 [   28 18287   195     5]
 [    0  1641   235     2]
 [    9  1037    29     8]]
```

Table:1 Confusion Matrix of Logistic Regression

	precision	recall	f1-score	support
1.0	0.74	0.15	0.24	705
2.0	0.85	0.99	0.91	18515
3.0	0.50	0.13	0.20	1878
4.0	0.53	0.01	0.01	1083
accuracy			0.84	22181
macro avg	0.66	0.32	0.34	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 0.842
- The testing prediction accuracy for Logistic Regression is 0.844
- Train mean squared error= 0.309
- Test mean squared error= 0.305
- Precision Average Score= 0.80

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

Because the error is constant across all the utilized solvers, including "newton cg," "lbfgs," "sag," and "saga," optimizing the penalty score has no impact on either the training model or the testing model.

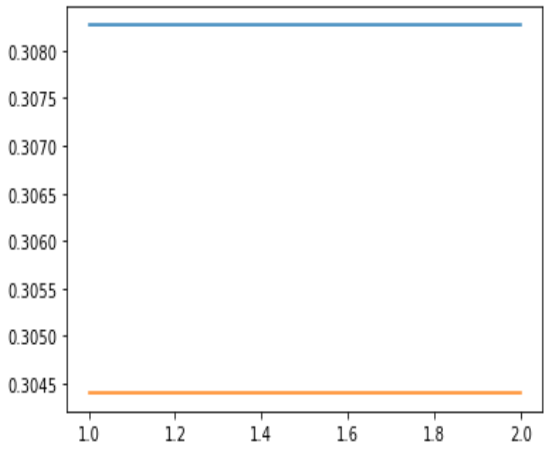

Figure 3: MSE for Logistic Regression training and testing with different penalty terms

2) Linear Discriminant Analysis:

Widely used in statistics, pattern recognition, and machine learning, the linear discriminant analysis is a generalization of Fisher's linear discriminant approach. Finding a linear combination of features that can distinguish between two or more classes of objects is the goal of the LDA technique. With LDA, class separability is maximized in the representation of data. While projection allows for the grouping of objects of the same class, it places objects of different classes far apart from one another.

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—"svd," "lsqr," and "eigen"—were employed. and the shrinkage was 0.0; the model's accuracy during training and testing was, respectively, 0.824 and 0.827. Precision data demonstrates that LDA significantly lowers type 1 error (actual positive observed out of the total positive results). The lowest Precision Accuracy Scores are for the Precision Accuracy. Below are references to the LDA classification report and confusion matrix.

```
[[  197   481    26     1]
 [  335 17585   435   160]
 [   23  1380   435    40]
 [   39   840    67   137]]
```

Table:3 Confusion Matrix of LDA

	precision	recall	f1-score	support
1.0	0.33	0.28	0.30	705
2.0	0.87	0.95	0.91	18515
3.0	0.45	0.23	0.31	1878
4.0	0.41	0.13	0.19	1083
accuracy			0.83	22181
macro avg	0.51	0.40	0.43	22181
weighted avg	0.79	0.83	0.80	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 0.824
- The testing prediction accuracy for LDA is 0.827
- Train mean squared error for LDA= 0.337
- Test mean squared error for LDA= 0.328
- Precision Average Score= 0.79

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

It was discovered that svd outperformed lsqr and eigen in the grid search solver (single value decomposition). produced the finest outcomes. It had a mean accuracy of 0.824. Shrinkage also provides no improvement; therefore, it was set to 0.0.

3) K-Nearest Neighbor:

The goal of nearest neighbor methods is to identify a set of training samples that are most nearby the new point and use them to predict the label. The quantity of samples may be an arbitrarily chosen constant (k-nearest neighbor learning). But depending on the number of nearby points and their density, it may change (radius-based neighbor learning). Any metric measure can be applied in general, although the most common one is the traditional Euclidean distance.

a) Explain all the parameters and hyperparameters:

- n_neighbors=10;34. As the name suggests, the number of neighbors is 10 and 34 for K-neighbors 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:

With 0.849 and 0.840, respectively, KNN has a little higher mean accuracy on training and testing than the other models, but we know this isn't the only aspect we're taking into account when choosing our best model. As a result, in addition to the training and testing errors, an average precision score for all the severity is found. This number is one of the best among the models. This model might have

low bias and variance due to how little the testing and training mistakes change from one another. Below is a reference to the KNN classification report and confusion matrix.

```
[[  152   535    14     4]
 [   73 18185   229    28]
 [    4  1581   280    13]
 [   13  1016    34    20]]
```

Table:5 Confusion Matrix of KNN

	precision	recall	f1-score	support
1.0	0.63	0.22	0.32	705
2.0	0.85	0.98	0.91	18515
3.0	0.50	0.15	0.23	1878
4.0	0.31	0.02	0.03	1083
accuracy			0.84	22181
macro avg	0.57	0.34	0.37	22181
weighted avg	0.79	0.84	0.79	22181

Table: 6 Classification Report for KNN

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

- The training prediction accuracy for KNN is 0.849
- The testing prediction accuracy for KNN is 0.840
- Train mean squared error for KNN= 0.295
- Test mean squared error for KNN= 0.305
- Precision Average Score= 0.79

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

Using GridSearchCV and cross val score, the ideal number of neighbors for the KNN model was determined to be 31. From the graph below, where the KNN model was fitted, n = 29 was selected as the value of k for KNN. Based on this value, the 10-fold cross-validation score was 0.844 (Training) and 0.842 (Testing).

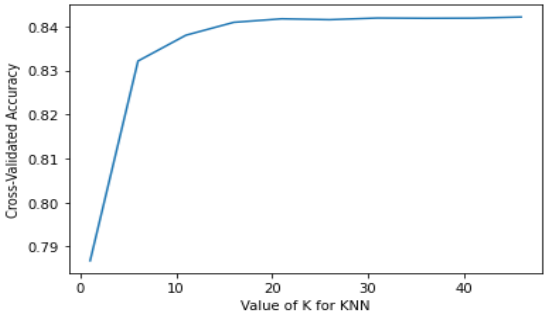

Figure 4: Value of K for KNN vs. Cross Validated Accuracy

4) Decision Tree Classifier:

The Decision Tree Classifier uses supervised machine learning to learn fundamental decision rules from training data to predict the target variables. This approach is easy to see as well as understand and interpret. This method is risky, though, as little adjustments to the data set could generate a completely different tree. Additionally, this technique is referred to as "greedy" because the best decisions are made locally at each node, hence it may not always provide the globally optimal decision tree.

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: 54 This determines the decision tree's maximum depth or expansion.Using RandomizedSearchCV between 0 and 100 and 'None,' the max depth was set at 54. If 'None' is used, the tree will grow until all leaves have fewer samples than the min samples split.
- Min_samples_split:89 , This is the minimum number of samples in the internal node after which it can be split. The min_samples_split was chosen as 61 using RandomizedSearchCV between the range 0 and 100.
- Min_samples_leaf:6,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: ‘auto’ 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.

b) Evaluating the performance of the models:

For training and testing, the Decision Tree Classifier's precision accuracy is 0.999 and 0.839, respectively. Consequently, the data have a large variance and little bias. However, the precision rating is 0.82. The Decision Tree Classifier's confusion matrix and classification report are provided below.

```
[[  336   342    27     0]
 [   82 18068   288    77]
 [   35  1337   495    11]
 [   12   887    83   101]]
```

Table:7 Confusion Matrix of Decision Tree

	precision	recall	f1-score	support
1.0	0.72	0.48	0.57	705
2.0	0.88	0.98	0.92	18515
3.0	0.55	0.26	0.36	1878
4.0	0.53	0.09	0.16	1083
accuracy			0.86	22181
macro avg	0.67	0.45	0.50	22181
weighted avg	0.83	0.86	0.83	22181

Table: 8 Classification Report for Decision Tree

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

- The training prediction accuracy for Decision Tree is 0.999
- The testing prediction accuracy for Decision Tree is 0.839
- Train mean squared error for Decision Tree = 0.285
- Test mean squared error for Decision Tree= 0.286
- Precision Average Score= 0.83

5) Random Forest:

With the Random Forest Classifier, numerous decision tree classifiers are fitted to different subsets of the data set. The target variable is forecast using all subgroups, and the final prediction results are then averaged

a) Explain all the parameters and hyperparameters:

- n_estimators: ‘700’ The number of trees in the forest.
- Min_samples_split:9, This is the minimum number of samples in the internal node after which it can be split. The min_samples_split was chosen as 9 using RandomizedSearchCV between the range 0 and 50.
- Min_samples_leaf:11,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.
- Max_features: ‘None’ The maximum number of features to consider when looking for the best split is None.
- Max_depth: 28 This determines the decision tree's maximum depth or expansion. Using Randomized Search CV between 0 and 80 and 'None,' the max depth was set at 28. If 'None' is used, the tree will grow until all leaves have fewer samples than the min samples split.
- Bootstrap: True, since this is the case, training samples can be of any day and contain any quantity of characteristics. So, for instance, it is possible to forecast the result of date (t+1) by using samples from days (t-15), (t-19), and (t-35) each with randomly chosen attributes.

b) Evaluating the performance of the models:

The random classifier, which has the highest precision accuracy score of 0.858, is the best-performing model. In terms of both training and testing data sets, the MSE also has the lowest value. Below are the Random Forest Classifier's confusion matrix and classification report.

```
[[  290   401    14     0]
 [   25 18397    92     1]
 [   18  1435   425     0]
 [   11  1009    45    18]]
```

Table: 9 Confusion Matrix of Random Forest

	precision	recall	f1-score	support
1.0	0.843	0.411	0.553	705
2.0	0.866	0.994	0.925	18515
3.0	0.738	0.226	0.346	1878
4.0	0.947	0.017	0.033	1083
accuracy			0.862	22181
macro avg	0.849	0.412	0.464	22181
weighted avg	0.858	0.862	0.821	22181

Table: 10 Classification Report for Random Forest

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

- The training prediction accuracy for Random Forest is 0.866
- The testing prediction accuracy for Random Forest is 0.862
- Train mean squared error for Random Forest = 0.281
- Test mean squared error for Random Forest= 0.282
- Precision Average Score= 0.858

d) Perform bias-variance trade-off for hyperparameter optimization of the Decision Tree and Random Forest:

Using a RandomizedSearchCV, the best parameters for the Decision Trees Classifier and the Random Forest Classifier were discovered. Decision trees have a low bias and a high variance since they tend to overfit the training data. By producing a large number of trees and averaging together a large number of weakly connected trees, Random Forests try to minimize variance by lowering the correlation between the trees and reducing overfitting. By experimenting with hyperparameters such max features, min samples leaf, and min samples split, the bias-variance trade-off was eventually discovered. By preventing the Decision Tree from branching out for every case, an increase in the minimum samples leaf low variance. If the minimum samples in the leaf node were lesser than 15, which was found to be the case for a random forest classifier, the model would not overfit the data. For min samples split, the same holds true.

Parameters	Decision Tress Classifier	Random Forest Classifier
Max features	Auto	Auto
Max depth	54	28
Min sample split	89	9
Min sample leaf	6	11
Criterion	entropy	Bootstrap: True
Model Score (Training)	0.84	0.86

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

6) Neural Network:

Deep learning techniques are based on neural networks, sometimes referred to as artificial neural networks (ANNs) or simulated neural networks (SNNs), which are a subset of machine learning. Their structure and nomenclature are modeled after the human brain, mirroring the communication between organic neurons.

A node layer of an artificial neural network (ANN) consists of an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, is connected to others and has a weight and threshold that go along with it. Any node whose output exceeds the defined threshold value is activated and begins providing data to the network's uppermost layer. Otherwise, no data is transferred to the network's next tier.

a) Explain all the parameters and hyperparameters:

- Number of Hidden Layers and units: Between the input layer and the output layer are layers known as hidden layers. Accuracy can be improved by using regularization techniques on numerous hidden units within a layer. Underfitting may result from a smaller number of pieces.

- Dropout: Dropout is a regularization approach that reduces overfitting (increases validation accuracy), hence boosting generalizability.
- Network Weight Initialization: The activation function used on each layer should determine which weight initialization approach should be employed. The distribution is mostly uniform.
- Activation function: Deep learning models can learn nonlinear prediction boundaries because activation functions are utilized to provide nonlinearity to models. The rectifier activation function is typically the most widely used. Binary predictions are made in the output layer using sigmoid. The output layer employs SoftMax to provide multi-class predictions.
- Learning Rate: How rapidly a network updates its parameters depends on its learning rate. Low learning rates cause the learning process to progress more slowly but smoothly. Though it might not converge, a higher learning rate accelerates learning. A declining learning rate is typically desired.
- Momentum: With the information of the past steps, momentum helps to choose the direction of the next step. It aids in stopping oscillations. The usual range for momentum is 0.5 to 0.9.
- Number of epochs: When training, the network is shown the entire training set a certain number of times, or epochs. Even while training accuracy is improving, increase the number of epochs until the validation accuracy starts to decline (overfitting).
- Batch size: The number of sub samples provided to the network as part of the mini batch size is when parameter updating takes place.32 could be a reasonable batch size default.

b) Evaluating the performance of the models:

For training and testing, the Neural Network accuracy is 0.85 and 0.83, respectively. Consequently, the data have a low variance and low bias. Also

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

- The training prediction accuracy for Neural Network is 0.85
- The testing prediction accuracy for Neural Network is 0.83
- The Validation prediction accuracy for Neural Network is 0.84
- Train mean squared error for Random Forest = 0.15
- Test mean squared error for Random Forest= 0.17

d) Perform bias-variance trade-off for hyperparameter optimization of the Neural Network

The bias-variance trade-off demonstrates that as model complexity rises, bias falls and variance rises. A U-shaped test error curve is the end outcome. However, the usual U-shaped test error curve is conspicuously absent from recent experimental results utilizing overparameterized neural networks. Large networks keep test errors down. This implies that there may not be a trade-off between bias and variance in neural networks with regard to network width.

The performance of algorithms is measured using interpretable accuracy metrics. Model correctness is often assessed based on model parameters and expressed as a percentage. This is a gauge of how closely the model's predictions match the observed data.

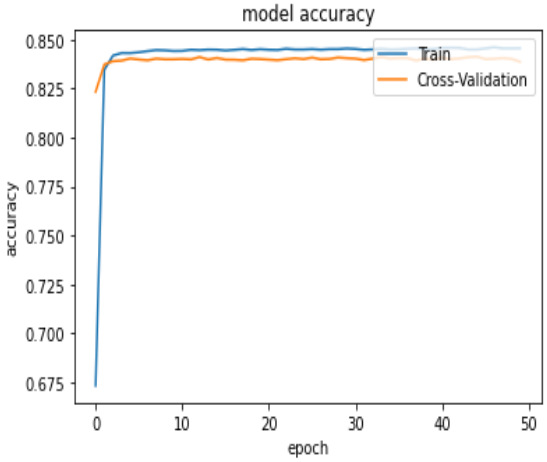

Figure:5 Model Accuracy of Neural Network

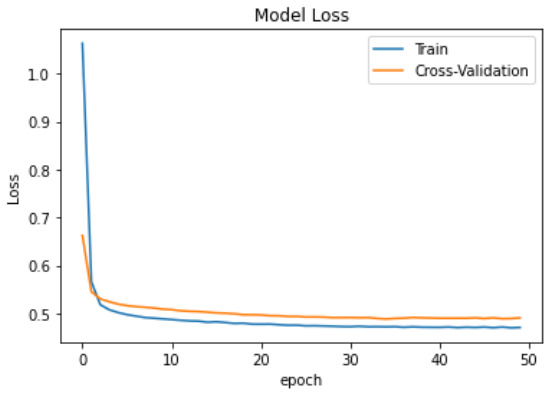

Figure:6 Model Loss of Neural Network

5. Model Selection.

a) Shrinkage Methodology:

1) Ridge Regression:

The data that exhibits multicollinearity can be analysed using the model tuning technique known as ridge regression. This technique carries out L2 regularization. Predicted values differ much from real values when the problem of multicollinearity arises, least-squares are unbiased, and variances are significant.

Ridge regression is a basic strategy for reducing model complexity and avoiding over-fitting caused by simple linear regression. R2 for ridge shrinking is 0.12 for both the training and test sets of data. Additionally, training and testing MSE are 0.258 and 0.255 respectively. Poor results from the Ridge Regression Model indicate that the data set is very non-linear.

b) Dimension Reduction Technique:

Dimensionality reduction is a method for lowering a data feature set's dimension. In a three-dimensional space, a machine learning dataset (feature set) typically consists of an array of points or hundreds of columns, or features. You can reduce the number of columns to quantifiable counts by using dimensionality reduction, which will reduce the three-dimensional sphere to a two-dimensional object (circle).

1) Principal component analysis:

One of the most popular linear dimensionality reduction approaches is principal component analysis. In order to optimize the variance of the data in the low-dimensional representation, this approach directly maps the data to a space with fewer dimensions.

In essence, it is a statistical process that orthogonally transforms a dataset's 'n' coordinates into a fresh set of n coordinates, referred to as the principal components. The first principal component that is produced as a result of this conversion has the most variance. If a principal component is orthogonal (i.e., not correlated) to those that came before it, it will have the largest variance imaginable.

In principle component analysis, the dimension of the original dataset is divided into 'n' principal components, which aids in reducing the multi-collinearity issue and improves the performance and predictive power of the model. The aforementioned figure demonstrates that 35 was selected as the best value for the number of primary components because it accounts for 90% of the variance.

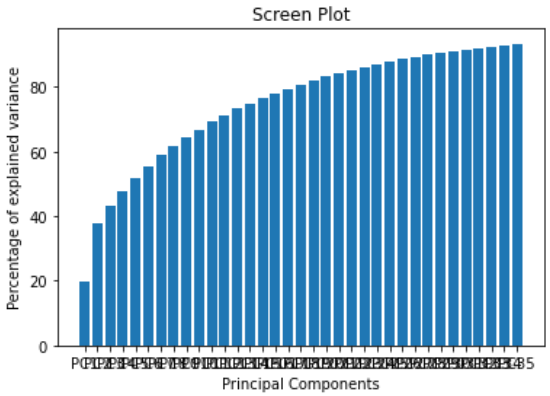

Figure:7 Hyperparameter optimization after PCA for Number of Principle Components

2) Partial Least Squares:

PLS regression is a statistical technique that resembles principal component regression in that it finds a linear regression model by projecting the predicted variables and the observable variables to a new space rather than searching for hyperplanes of maximum variance between the response and independent variables. The PLS family of techniques are referred to as bilinear factor models because both the X and Y variables are projected to new spaces. When the Y is categorical, a version called partial least squares discriminant analysis (PLS-DA) is employed.

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

The ideal hyperparameters of the data set with less features were found using Random Forest in the same way. To discover the best hyperparameters for the data set with less features, RandomizedSearchCV was employed. This model with enhanced hyperparameters was fitted to the smaller dataset and evaluated similarly to the prior scenario.

c) Best overall model

The Random Forest Classifier was chosen as the best of the six fitted models. As already mentioned, the essential metrics for assessing

all fitted models were precision and the difference between training and testing errors. Furthermore, the difference in errors was greater than with Random Forest, despite the fact that K-Nearest Neighbors and Logistic Regression had higher accuracy ratings among the fitted models. This model has a low bias compared to the others because the training and testing accuracy differences are so small and the training accuracy is 0.866, which is regarded as an outstanding match. Additionally, the testing accuracy is 0.862, which is quite similar to the training accuracy, showing that the model is effective in explaining variation in the testing data-set is low variance than other models. In addition, the average accuracy rating is really good.

d) Comparing the model findings to the correlation matrix results

Full Dataset	Reduced Dataset using PCA
Mean Accuracy (Training):0.866	Mean Accuracy (Training):0.946
Mean Accuracy (Testing):0.862	Mean Accuracy (Testing):0.846
Precision Avg.Score:0.857	Precision Avg.Score:0.812

Table 12: Comparison between the results from Random Forest and Reduced dataset after PCA

e) Interpreting the findings and comparing against the observation:

We can see that the model's performance in terms of the bias-variance trade-off and accuracy average score has not improved after being fitted with the best hyperparameters for the smaller dataset.

The model's performance in predicting the testing data has not increased because the discrepancy between the training and testing accuracies is bigger in the reduced dataset. When compared to the previous model, the bias and variance of this model are large. This indicates that the majority of the lost features are not projected or translated onto the smaller set of primary components. The model may not have done well in properly forecasting more True Positive values, which ultimately increased the False Positive, as evidenced by the reduction in the precision average.

6.Conclusion

Six different machine learning models based on several functional areas were used to predict the severity of accidents. A slight multicollinearity problem between the predictors was found using the correlation matrix and heat map, but otherwise the predictors were independent. All five machine learning models had equivalent performance. In spite of this, Random Forest surpassed the competition in every performance assessment metric, including accuracy, precision score, and training and testing mistakes.

To get the greatest model performance, the Random Forest hyperparameters were improved using the bias-variance trade-off before being fitted to the data set. Due to the very modest difference between training and testing errors and accuracies, this model demonstrated low bias and variance in comparison to other models. Additionally, it was observed that the precision score rose to 0.86, indicating that the model correctly predicted more True Positive values while lessening the number of False Positive values.

To further reduce the multi-collinearity issue and improve the model's performance, PCA was utilized as a dimensionality reduction technique. 35 principal components were created by condensing 98 features, and they were then utilized to explain the variation in the dataset. Since the reduction is not very high and the number of PCA dimensions is not at all low,

we may conclude that multi-collinearity has little to no influence.

The model's performance in predicting the testing data has not increased because the discrepancy between the training and testing accuracies is bigger in the reduced dataset. When compared to the previous model, the bias and variance of this model are large. This indicates that the majority of the lost features are not projected or translated onto the smaller set of primary components. The model performed poorly in properly forecasting more True Positive values, which ultimately increased the False Positive values. The accuracy average has also decreased to 0.812, which implies that the model fared poorly in doing so.