**COMP 247 Supervised Learning**

**KSI Dataset – Analysis Report**

**Group Members:**

**Alakh Patel – 301232383**

**Dhrumit Patel – 301171388**

**Gitansh Mittal – 301200517**

**Yagnam Sukhadia -**

**Executive Summary**

The goal of the research is to create a machine learning model that uses the KSI (Killed or Seriously Injured) dataset to estimate the probability of a fatal injury in a traffic collision. Information on traffic accidents that caused fatalities or significant injuries is included in the dataset. The project went through a number of phases, including feature selection, data exploration, and model creation. The final model's accuracy rate of 96% suggests that it can be helpful in foretelling the probability of a fatal injury in a traffic collision.

**Overview of Solution**

To find a solution, a classification algorithm was used to create a machine learning model that could forecast the probability of a fatal injury in a traffic collision. The model considers a number of variables, such as location, road type, weather, and vehicle type, among others. To assure the model's accuracy, the research also entailed data cleaning and preprocessing. The final model's accuracy rate of **96%** suggests that it can be helpful in foretelling the probability of a fatal injury in a traffic collision.

**Data Exploration and Findings**

Information on traffic accidents that resulted in fatalities or serious injuries can be found in the KSI dataset. There are 57 characteristics in the dataset. To analyse and display the dataset, we used the Python programming language and modules like Pandas, NumPy, and Matplotlib. We discovered that the dataset contained missing values, therefore we cleaned the data using methods like imputation and removing columns with an excessive number of missing values.

**ADDING A TABLE .CSV, ADDING GRAPHS, ADDING PROFILING IN NAV BAR**

**Feature Selection**

We used several techniques such as correlation analysis, mutual information, and analysed the null values to select the most relevant features for our model. We found that some of the most important features for our model were location, road type, weather conditions, and vehicle type.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Field Name** | **Description** | **Selected or not? (Reason)** |
| 1 | INDEX\_ | Unique Identifier | Dropped |
| 2 | ACCNUM | Accident Number | Dropped |
| 3 | YEAR | Year Collision Occurred | Dropped |
| 4 | DATE | Date Collision Occurred | Dropped |
| 5 | TIME | Time Collision Occurred | Selected |
| 7 | STREET1 | Street Collision Occurred | Dropped |
| 8 | STREET2 | Street Collision Occurred | Dropped |
| 9 | OFFSET | Distance and direction of the Collision | Dropped |
| 10 | ROAD\_CLASS | Road Classification | Selected |
| 11 | DISTRICT | City District | Selected |
| 12 | WARDNUM | City of Toronto Ward collision occurred | Dropped |
| 13 | LATITUDE | Latitude | Dropped |
| 14 | LONGITUDE | Longitude | Dropped |
| 15 | LOCCOORD | Location Coordinate | Selected |
| 16 | ACCLOC | Collision Location | Dropped |
| 17 | TRAFFCTL | Traffic Control Type | Selected |
| 18 | VISIBILITY | Environment Condition | Selected |
| 19 | LIGHT | Light Condition | Selected |
| 20 | RDSFCOND | Road Surface Condition | Selected |
| 21 | ACCLASS | Classification of Accident | Label CLass |
| 22 | IMPACTYPE | Initial Impact Type | Dropped |
| 23 | INVTYPE | Involvement Type | Dropped |
| 24 | INVAGE | Age of Involved Party | Selected |
| 25 | INJURY | Severity of Injury | Selected |
| 26 | FATAL\_NO | Sequential Number | Dropped |
| 27 | INITDIR | Initial Direction of Travel | Dropped |
| 28 | VEHTYPE | Type of Vehicle | Dropped |
| 29 | MANOEUVER | Vehicle Manouever | Dropped |
| 30 | DRIVACT | Apparent Driver Action | Selected |
| 31 | DRIVCOND | Driver Condition | Selected |
| 32 | PEDTYPE | Pedestrian Crash Type - detail | Selected |
| 33 | PEDACT | Pedestrian Action | Dropped |
| 34 | PEDCOND | Condition of Pedestrian | Dropped |
| 35 | CYCLISTYPE | Cyclist Crash Type - detail | Dropped |
| 36 | CYCACT | Cyclist Action | Dropped |
| 37 | CYCCOND | Cyclist Condition | Dropped |
| 38 | PEDESTRIAN | Pedestrian Involved In Collision | Selected |
| 39 | CYCLIST | Cyclists Involved in Collision | Selected |
| 40 | AUTOMOBILE | Driver Involved in Collision | Selected |
| 41 | MOTORCYCLE | Motorcyclist Involved in Collision | Selected |
| 42 | TRUCK | Truck Driver Involved in Collision | Selected |
| 43 | TRSN\_CITY\_VEH | Transit or City Vehicle Involved in Collision | Selected |
| 44 | EMERG\_VEH | Emergency Vehicle Involved in Collision | Selected |
| 45 | PASSENGER | Passenger Involved in Collision | Selected |
| 46 | SPEEDING | Speeding Related Collision | Selected |
| 47 | AG\_DRIV | Aggressive and Distracted Driving Collision | Selected |
| 48 | REDLIGHT | Red Light Related Collision | Selected |
| 49 | ALCOHOL | Alcohol Related Collision | Selected |
| 50 | DISABILITY | Medical or Physical Disability Related Collision | Selected |
| 51 | HOOD\_158 | Unique ID for City of Toronto Neighbourhood (new) | Dropped |
| 52 | NEIGHBOURHOOD\_158 | City of Toronto Neighbourhood name (new) | Dropped |
| 53 | HOOD\_140 | Unique ID for City of Toronto Neighbourhood (old) | Dropped |
| 54 | NEIGHBOURHOOD\_140 | City of Toronto Neighbourhood name (old) | Dropped |
| 55 | DIVISION | Toronto Police Service Division | Dropped |
| 56 | ObjectID | Unique Identifier (auto generated) | Dropped |

**Data Modeling**

To create our model, we used a classification technique. First, we used an 80:20 ratio to divide the data into training and testing sets. To make sure that all features were scaled to the same level, we scaled the data using the Scikit-Learn library's StandardScaler function. To overcome class imbalance, we also used a variety of data processing techniques, like oversampling(SMOTE).

**Model Building**

To create our model, we experimented with a number of techniques, including Logistic Regression, Random Forest, Naive Bayes - Bernoulli, Neural Network (MLP Classifier), and Support Vector Machine (SVM). In the test data, we discovered that the SVM algorithm performed the best, achieving an accuracy of **\_\_%.** In order to assess the effectiveness of our model, we also employed a confusion matrix, which revealed that it had a high true positive rate and a low false negative rate, suggesting that it could be helpful in forecasting the chance of a fatal injury in a traffic collision.

**Models Created:**

For the basic parameters (without fine-tuning):

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Cross-Val score (Mean) | Accuracy Score | |
|  | Train | Train | Test |
| SVM | 0.9133 | 0.9313 | 0.9149 |
| Logistic Reg. | 0.7826 | 0.7859 | 0.7778 |
| Random Forest | 0.9597 | 1.00 | 0.9620 |
| Naïve Bayes - Bernoulli | 0.80639 | 0.8075 | 0.8023 |
| Neural Network (MLP Classifier) | 0.80573 | 0.8151 | 0.8049 |

Fine-tuning the models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Best Parameters | Best Estimator | Accuracy Score | Confusion Matrix |
| SVM | {'svc\_\_C': 100, 'svc\_\_degree': 2, 'svc\_\_gamma': 0.03, 'svc\_\_kernel': 'rbf'} | Pipeline(steps=[('svc',  SVC(C=100, degree=2, gamma=0.03, probability=True,  random\_state=42))]) | 94.1257 | [[4315 186]  [ 364 4163]] |
| Logistic Reg. | {'log\_\_C': 10, 'log\_\_penalty': 'l1', 'log\_\_solver': 'liblinear'} | Pipeline(steps=[('log',  LogisticRegression(C=10, max\_iter=10000, penalty='l1',  solver='liblinear'))]) | 0.78167 | [[3650 851]  [1120 3407]] |
| Random Forest | {'rf\_\_ccp\_alpha': 0.0, 'rf\_\_criterion': 'gini', 'rf\_\_max\_depth': 8, 'rf\_\_max\_features': 'log2', 'rf\_\_max\_leaf\_nodes': None, 'rf\_\_min\_samples\_leaf': 1, 'rf\_\_min\_samples\_split': 2, 'rf\_\_n\_estimators': 100} | Pipeline(steps=[('rf',  RandomForestClassifier(max\_depth=8, max\_features='log2'))]) | 0.8845 | [[4108 393]  [ 649 3878]] |
| Naïve Bayes - Bernoulli | {'nb\_\_alpha': 0.5, 'nb\_\_binarize': 0.0} | Pipeline(steps=[('nb', BernoulliNB(alpha=0.5))]) | 0.80239 | [[3770 731]  [1053 3474]] |
| Neural Network (MLP Classifier) | {'nn\_\_activation': 'relu', 'nn\_\_hidden\_layer\_sizes': (12, 8), 'nn\_\_max\_iter': 100} | Pipeline(steps=[('nn',  MLPClassifier(hidden\_layer\_sizes=(12, 8), max\_iter=100,  random\_state=42))]) | 0.84625 | [[3884 617]  [ 771 3756]] |

**ROC Curves:**

**Support Vector Machine Classifier:**

Chart, line chart

Description automatically generated

**Logistic Regression:**

Chart, line chart

Description automatically generated

**Random Forest:**

Chart

Description automatically generated

**Naïve Bayes (Bernoulli):**

Chart, line chart

Description automatically generated

**Neural Network (MLP Classifier):**

Chart

Description automatically generated