ACCIDENT DATA ANALYSIS USING MACHINE LEARNING

Under The Guidance Of:
Sk. Riyaz Bagban

Designed By:
Sk. Aiman Sabaha
N. Lakshmi Sudheshna

Accident Data Analysis using Machine Learning

1. Project Overview

- **Objective**: The main purpose of the project is to have a user-friendly environment to analyse the accident data. This predicts severity of the accident that took place.
- **Scope**: The project covers data cleaning, visualization, modelling, and interface development using various ML algorithms and python methods.
- **Dataset Description**:
    - Kaggle
    - Key attributes and their descriptions:

      Accident Date: The date on which accident took place.

      Day of week: The day on which accident happened.

      Junction Details: It describes the spot of accident where it occur.

      Accident severity: Shows the severity of the accident.

      Light Condition: It shows the day light of the accident occur.

      Local Authority District: Police Force that took the report.

      Carriageway Hazards: accident comes under which category.

      No. of Casualties: Persons injured or died in that accident.

      No. of Vehicles: No. of Vehicles that involved in accident.

      Road surface Condition: It shows the condition of road (dry/wet).

      Road Type: It comes under one-way or two-way road.

      Speed Limit: Notes the speed limit of vehicle.

      Rural or Urban: Categorize the accident-prone area.

      Weather Condition: Shows the weather condition at the time of accident.

      Vehicle Type: Notes the type of vehicle involved in the accident.

- Tabular dataset:

  30429 x 21 size.
- Large dataset for better training & testing.

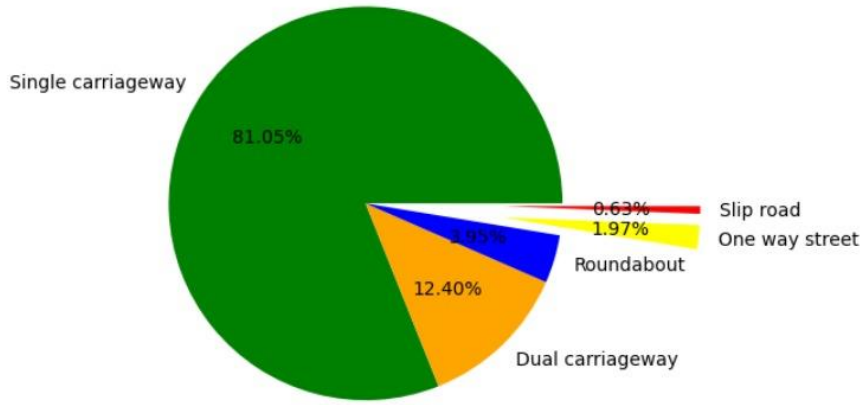
2. Data Cleaning and Preprocessing

- **Initial Analysis**:
  - Handling missing values.
  - Identifying and managing outliers.
  - Addressing duplicates.
- **Transformations**:
  - Data normalization or standardization.
  - Encoding categorical variables.
  - Feature selection and engineering: XGBoost with 80% accuracy

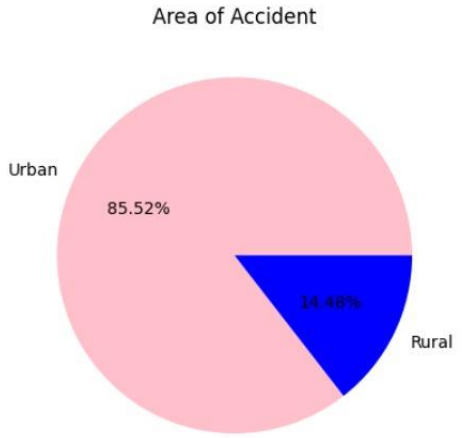
3. Exploratory Data Analysis (EDA)

- **Data Visualization:**
  - *Distribution plots for features.*

```
a=df['Road_Type'].value_counts()
explode=[0.7 if i==4 or i==3 else 0 for i in range(len(a))]
plt.pie(a,labels=['Single carriageway','Dual carriageway','Roundabout','One way street',
        'Slip road'],autopct='%.2f%%',colors=['green','orange','blue','yellow','red'],explode=explode);
```



```
plt.pie(df.Urban_or_Rural_Area.value_counts(),labels=['Urban','Rural'],autopct='%.2f%%',colors=['pink','blue'])
plt.title("Area of Accident");
```



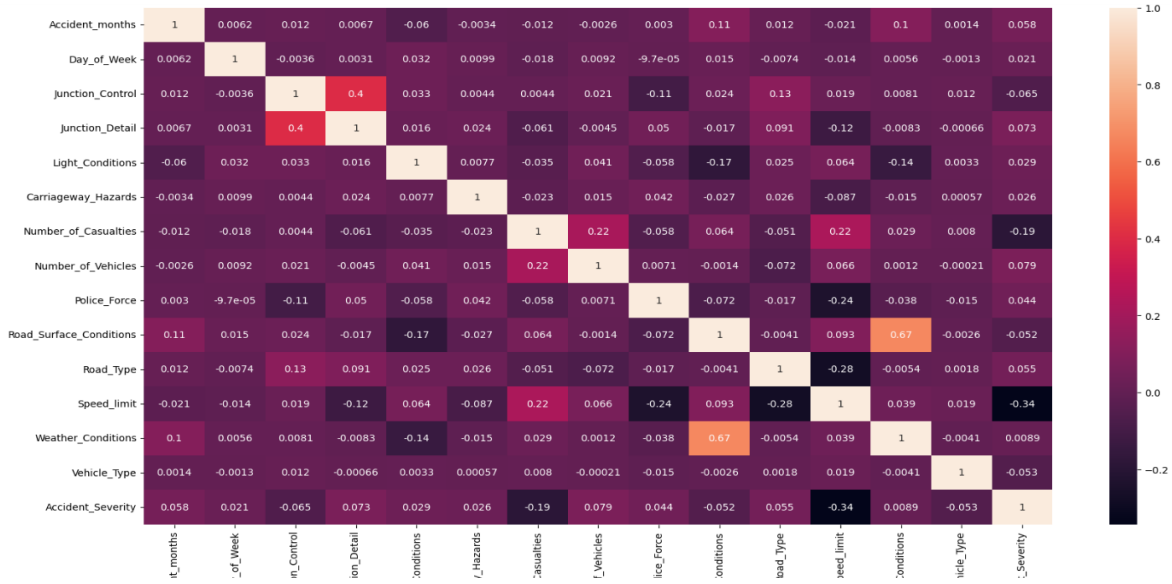

```
piv_tab=pd.pivot_table(df,index='Light_Conditions',columns='Accident_Severity',values='Number_of_Casualties',aggfunc=sum);
piv_tab
```

Accident_Severity Light_Conditions	Fatal	Serious	Slight
Darkness - lighting unknown	34	49	73
Darkness - lights lit	1033	2324	7258
Darkness - lights unlit	17	32	58
Darkness - no lighting	797	313	335
Daylight	2576	5292	19383

```
df.groupby('Carriageway_Hazards')[['Number_of_Casualties']].count()
```

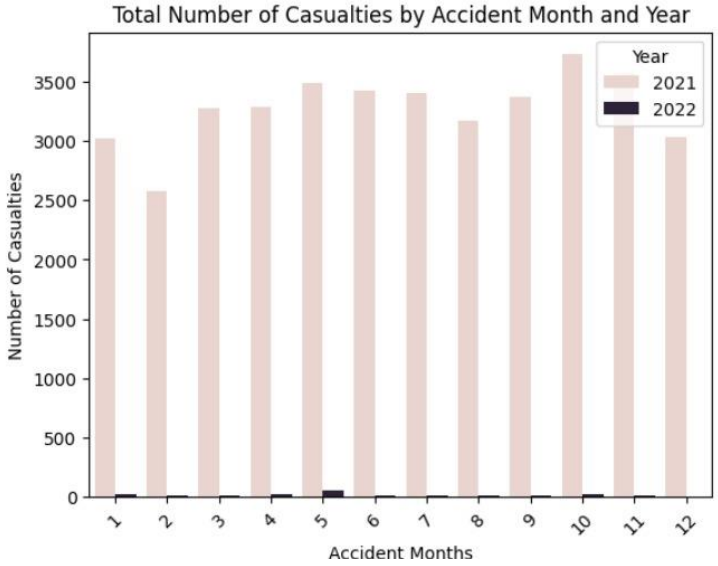
Carriageway_Hazards	Number_of_Casualties
Any animal in carriageway (except ridden horse)	79
None	30088
Other object on road	161
Pedestrian in carriageway - not injured	54
Previous accident	36
Vehicle load on road	11

- *Correlation heatmaps.*

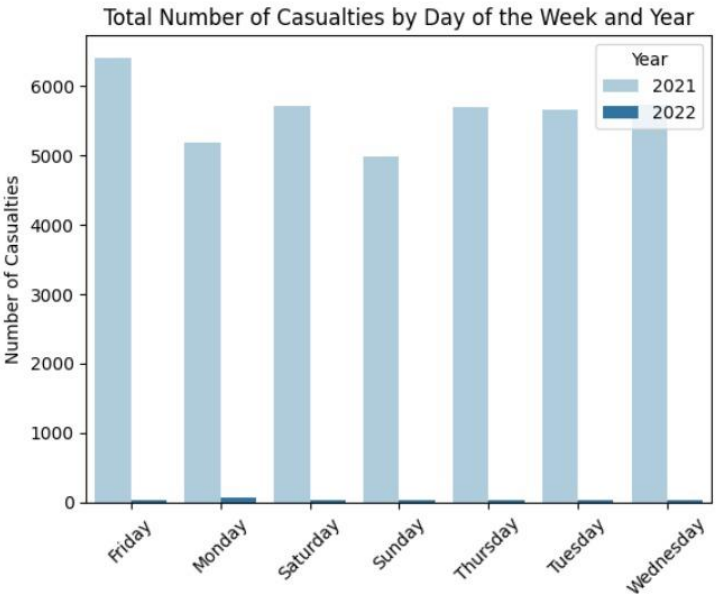


- *Relationship analysis using bar plot.*

```
df['Accident_Year'] = df['Accident_Date'].dt.year
# Aggregate the data by month and year
monthly_data = df.groupby(['Accident_months', 'Accident_Year'])['Number_of_Casualties'].sum().reset_index()
# Plot using Seaborn
sns.barplot(data=monthly_data, x='Accident_months', y='Number_of_Casualties', hue='Accident_Year')
plt.xlabel('Accident Months')
plt.ylabel('Number of Casualties')
plt.title('Total Number of Casualties by Accident Month and Year')
plt.xticks(rotation=45)
plt.legend(title='Year')
plt.show()
```




```
daily_data = df.groupby(['Day_of_Week', 'Accident_Year'])['Number_of_Casualties'].sum().reset_index()
sns.barplot(data=daily_data, x='Day_of_Week', y='Number_of_Casualties', hue='Accident_Year', palette='Paired')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Casualties')
plt.title('Total Number of Casualties by Day of the Week and Year')
plt.xticks(rotation=45)
plt.legend(title='Year')
plt.show()
```



4. Algorithm Implementation

- **Algorithms Applied**:
    - Classification is used to check and find the algorithm that provides more accuracy with the given values.
        - Logistic Regression.
        - Decision Tree Classifier.
        - Random Forest Classifier.
        - K Nearest Neighbors (KNN).
        - Gaussian NB.
        - XGBoost.
        - Bagging Classifier.
        - Support Vector Machine

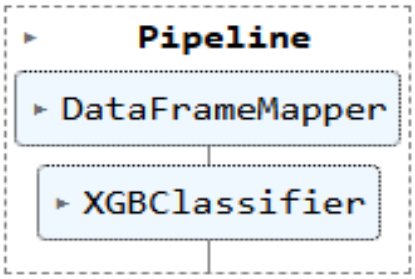
- **Model Training and Testing**:
    - Train-test split and cross-validation methods:

      Train-test split uses train_test_split for categorizing the target and features. KStratified-Fold is used for cross-validation. It acts well with imbalance data.
    - Accuracy and Prediction, Standard Deviation.
- **Accuracy Prediction**:
    - Results from various models are collected and compared so that the best model is selected for further process.
    - Comparative analysis of model performance.
    - Among the models XGBOOST algorithm is given better accuracy of 85.8%

5. Pipeline Development

- **Pipeline Design**: Outline the sequence of processes, including data preprocessing, model training, and prediction.

```
model=Pipeline(ml)
model.fit(X,y)
```



- Automated the workflow, from preprocessing to prediction ensuring reproducibility and efficiency.

6. User Interface (UI)

- **UI Design**:
  - The purpose of the interface is inputting new data and displaying predictions.
  - Streamlit is the Tool used in this Interface.

- **Functionality**:
  - The user opens the webpage and enter the accident details like Junction details, casualties, light condition, vehicle type, speed limit and road conditions in the section.
  - After entering the input, the user hits the Predict button just below the input section. This results in the output that displays the severity of the accident based on the algorithm selected.

7. Results and Evaluation

- **Final Model Performance**: Metrics and interpretation.
- **Visualization of Results**: Graphical representations for clarity.
- **Discussion**: As the dataset is large, the columns with additional information is added for better understanding and accuracy. Cleaning the data took a lot of time and effort for better prediction results.

8. Conclusion and Future Work

The project successfully showcased the complete data analysis workflow, including cleaning, visualization and prediction using machine learning algorithms. As our data is very much imbalanced, we tried to get the better accuracy as much as possible. It shows the results by selecting the best model, automating the processes with pipelines and creating a user-friendly interface seamless input and prediction.

Challenges:

Missing Values and Inconsistent Format

- The missing values were handled using imputation techniques like predictive methods.
- Data normalization and standardization ensures the consistency across features.

Algorithm Prediction

- Multiple algorithms were evaluated to find the best-performing one based on the accuracy and efficiency.
- Balancing the model accuracy with computational efficiency was achieved by optimizing algorithms and fine-tuning hyperparameters.

Pipeline Integration

- Built pipelines using frameworks like scikit-learn to automate the end-to-end process by standard scalar and Label Encoding.
- Modularize the pipeline for scalability and easy updates.

UI Design

- Designed the UI with intuitive workflows using Streamlit. Ensured the clear and responsiveness of results.

Future Scope:

- The interface can be further enhanced with more features, such as advanced visualizations or additional prediction models.
- Additional datasets and more complex models could be explored to further enhance accuracy and robustness.
- Due to imbalanced data, the predictions of model are inaccurate. For that you can use reinforcement learning to make accurate predictions.

9. Appendices

- Code snippets.
- Detailed tables or raw results.
- References for external resources or tools.
- For dataset: https://www.kaggle.com
- For Streamlit Documentation: https://docs.streamlit.io/
- For Pipeline Development: https://scikit-learn.org/stable/modules/pipeline.html
- For Visualization: https://plotly.com/python/