# 6103 Data Mining – Group Project

**Group 8**

**Individual Final Report: Jichong Wu**

## 1. Introduction

Thousands of vehicles involved rollovers in fatal traffic crashes every year. The rollover is one of the significant safety problems in fatal crashes. The National Highway Traffic Safety Administration (NHTSA) indicated that 18.9 percent fatal crashes in 2014 (7,592 of 40,164) involved rollovers. Occupants in a rollover crash have greater likelihood of experiencing fatal injuries than occupants in non-rollover crash. Reducing rollovers in traffic crashes will decrease fatalities. The purpose of this project is to predict rollovers in fatal single-passenger-vehicle crashes.

**Outline of Teamwork Responsibilities:**

- Jia-ern Pai: data cleaning, KNN model, and Logistic Regression model
- Ethan Litman: data cleaning, Decision Tree model and Random Forest model
- Jichong Wu: Random Forest model and GUI design

## 2. Description of My Individual Work:

1) All the GUI design and code;
2) Random Forest algorithm and code
3) Understanding and modifying some of team members codes to make them fit into PyQt5 environment, for example
    a. All the plots made by Seaborn and Pandas functions didn't work in PyQt5 and I converted to Matplotlib
    b. Our team member used statsmodels.api package as the classifier for Logistic Regression model and I converted the codes to LogisticRegression classifier in the sklearn.linear_model package and to get the same results my team mate did.
4) Detailed work and my contribution can be found below. It has the following key areas and challenges which I have overcame in the end.
    a. Learning GUI design from scratch and understanding all the basics, how to create windows, how to link windows, how to pass variable values between windows, how to design the layout of each window and how to create each element of a window – text box, button, dropdown list, graphs, etc.
    b. Converting codes from other packages (Seaborn, Pandas) of drawing plots to Matplotlib, and modify and customize the plots
    c. Making user input and selections on window dynamic with results in text and plots.

d. Customize all the elements in GUI, it took a lot of time and I learned all the nice tricks including text alignment, background color, font size and color, etc.

### 3 & 4. Detailed Individual Work and Results which was submitted in the Team Report:

# :: ========================
# :: ========================

## GUI Design

We used PyQt5 for this GUI design and code, and Qt Designer wasn't used.

## A. Overall Layout

The GUI is comprised of a main window and four dropdown menus:

- **About** – introduce the team, the professor of this class, and the Exit button
- **EDA** – display basic analysis and steps during data preprocessing and cleaning
- **Models** – display model metrics, results with charts by four different models we built, including Decision Tree, Random Forest, Logistic Regression and KNN
- **Conclusion** – display the accuracy scores of each model and note that we didn't make conclusion only based on the accuracy score, this is more for a good way of GUI to present virtually.


## B. The main window design

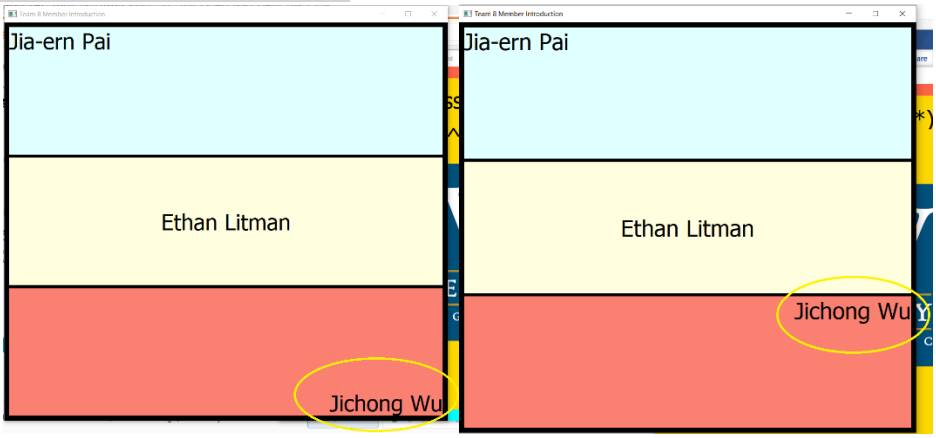
1) **Background**: comprised of some opening text messages (by QLabel), customized by setStyleSheet() function for text size and color. Also used the QPixmap() function of loading a photo to the window which was the main challenge for this part. The issue was

how to load a photo from the internet instead of local machine. The urllib.request.urlopen().read() did the job.

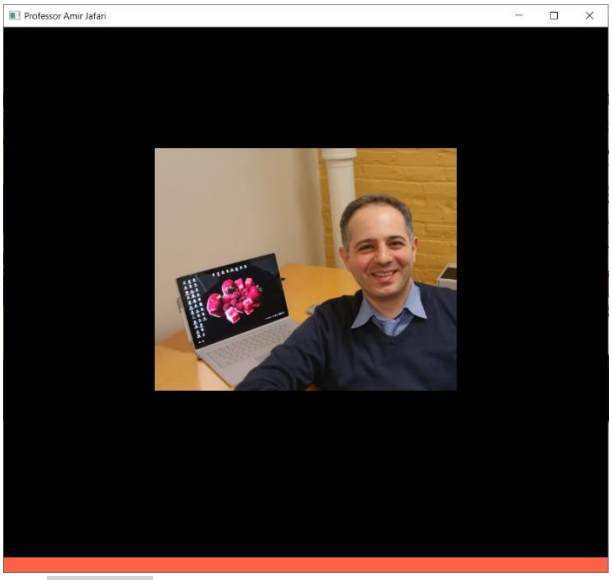
2) **Menu bar**: background color was consumed.
3) **Status tip bar**: background color and messages displayed in this section was customized.

### C. The "About" menu

1. **The "Team 9 Member Introduction" tab**: used QLabel to display team member names, adjusted the label size and background color; adjust the text of team member names to make it bold and big, also did some research on how to align the text within each label text box to make them line up nicely (top left, center, bottom right) by using setAlignment(QtCore.Qt.AlignRight/Center/Top)



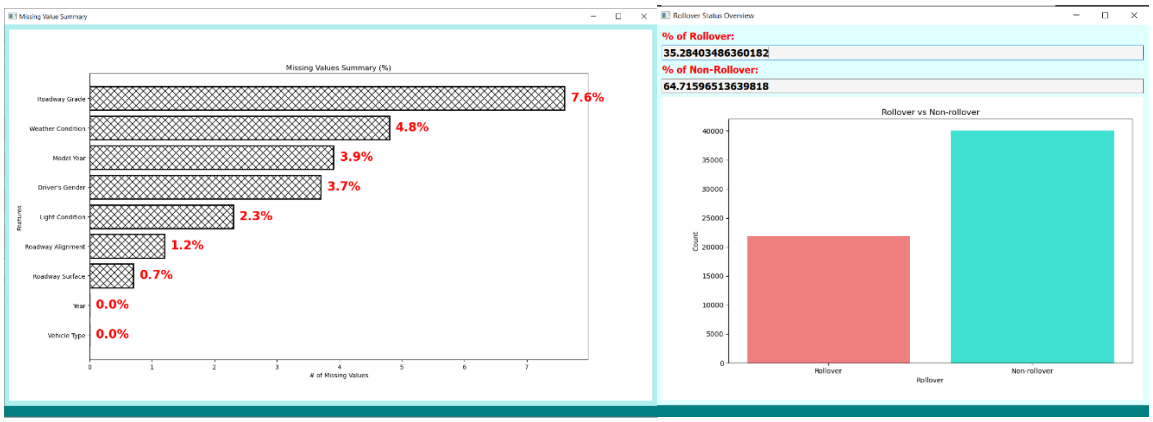

2. **The "Professor Amir Jafari" tab**: applied the QPixmap() and urllib.request.urlopen().read() technique used in the main window photo display. Photo of Professor Jafari credit goes to his GitHub page.




3. **The "Exit" button**: used QAction function for creating the exit button and linked with the triggered.connect() function to execute the closing window action.

## D. The "EDA" menu

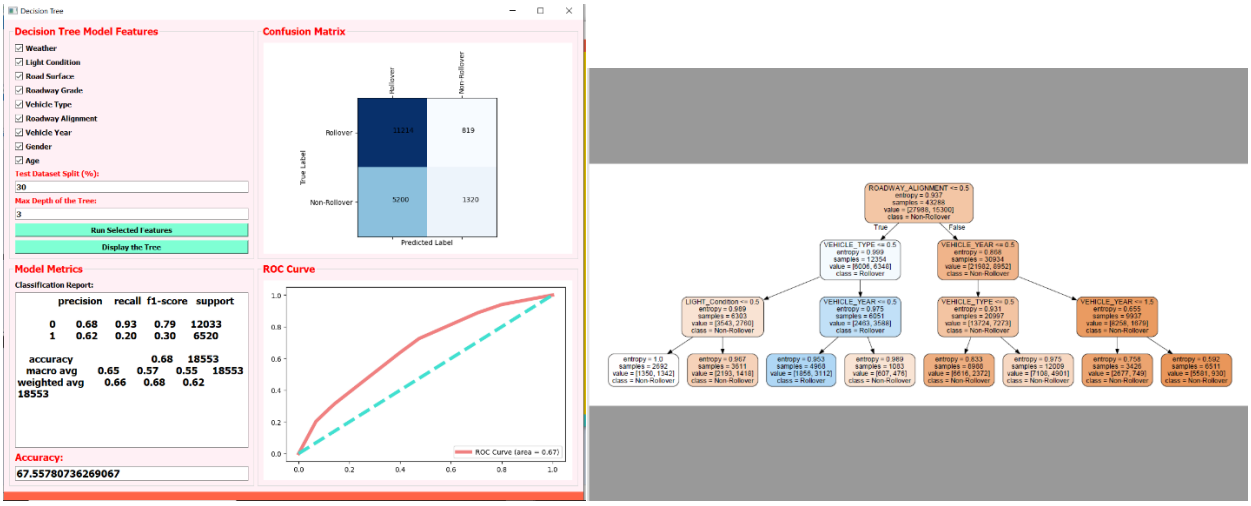
1) **The "Missing Values" tab**: use a horizontal bar plot to display, the number results for each bar are also displayed on top of each bar.



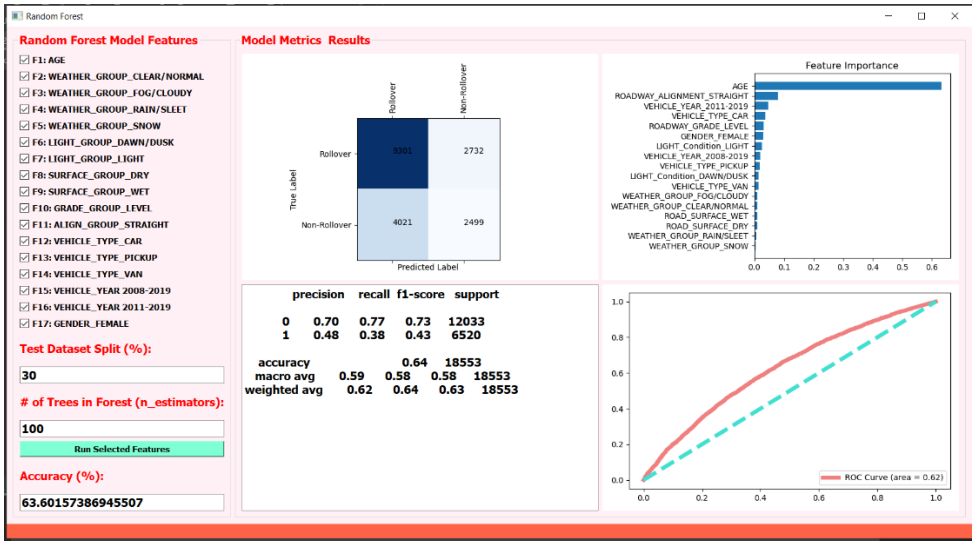

2) **The "Rollover Status Overview" tab**: displays the frequency count of the two Classes (rollover or non-rollover) in the Target variable (Rollover) by a bar chart and displays, QLabel and QLineEdit and setText() function were used to display the results.
3) **The "Rollover (Target) vs Features" tab**: displays the frequency count and percentage comparing against the Target variable. For the Age variable, we showed different techniques to treat Age as both a categorical variable (by grouping age segments) and a numerical variable. For numerical variable, we displayed not only bar plot which is not ideal for continuous data, and also created a stand alone checkbox and link to another (3rd layer) window to display a boxplot for Age vs Y variable, which is more appropriate for numerical variable.

## E. The "Models" menu

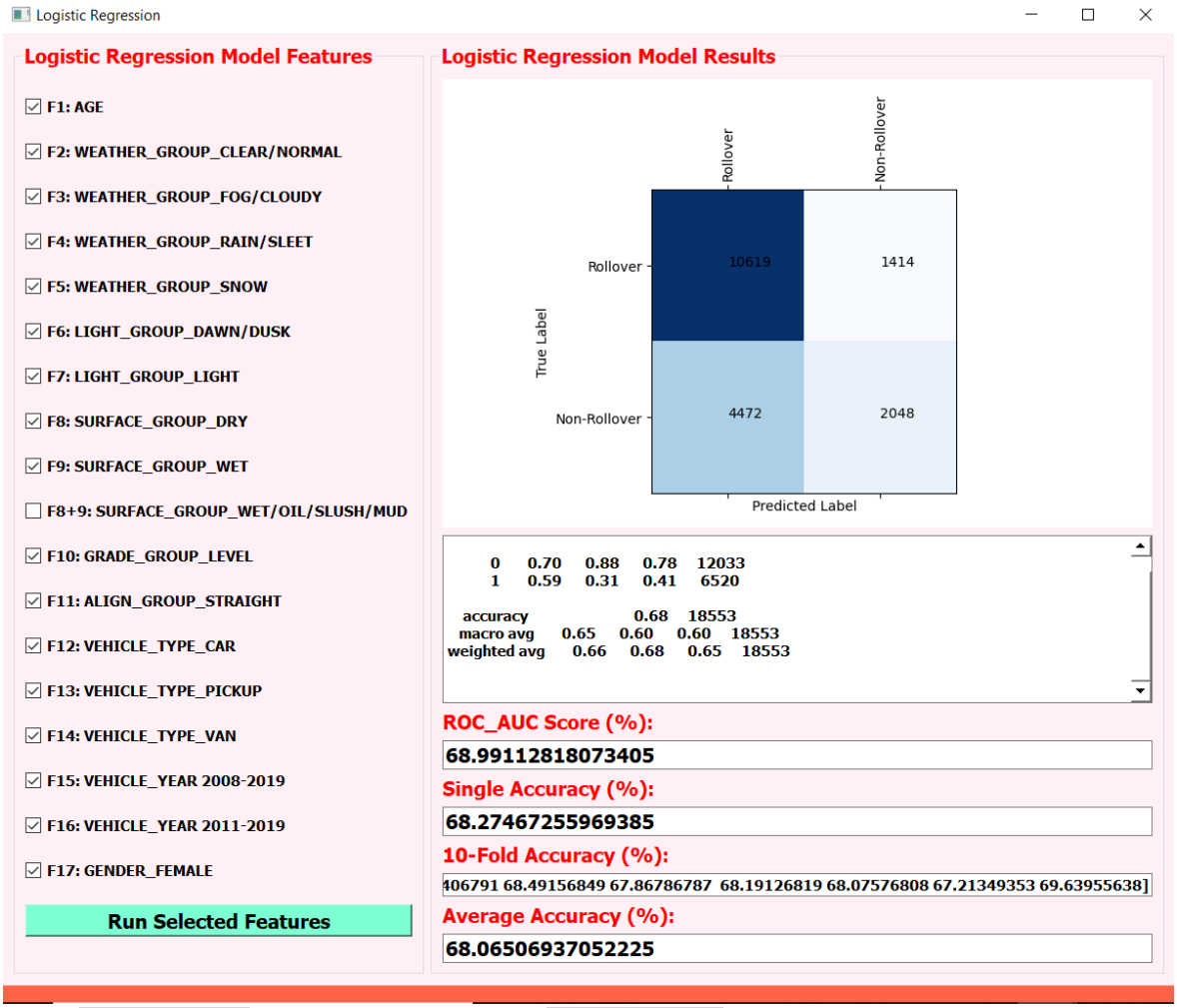
1) **Decision Tree**: layout was split into QGridlayout design which allows features and control buttons and to display on top left corner. Checkbox was used to select the features and it's linked to the display results. QPlainTextEdit() was used to create the classification report text box, and two graphics used for the confusion matrix plot and the ROC Curve. Finally a QPlainTextEdit() was used to display the Accuracy score of the model.



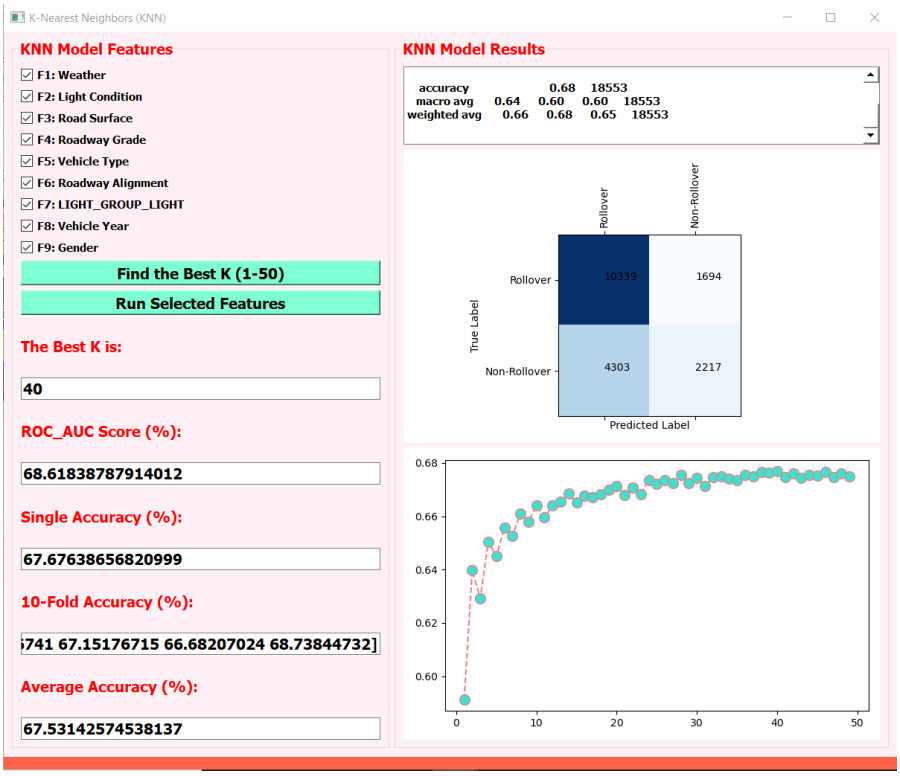

2) **Random Forest**: the layout design was similar to Decision Tree, except it used HBoxlayout and for the right area within the HBox, a QGridlayout design was used to create 4 graphics within the HBoxlayout as a second layer. The other unique thing for the Random Forest model is to take input for the n_estimators of the model. On the features, we used OneHotEncoding to split those features that have multiple labels, this is for the feature importance selection and our model will pick the first 15 important features.



3) **Logistic Regression**: same QHBoxlayout design was used, this time on the right side of the HBox, VBoxlayout was applied to accommodate the ROC_AUC score, the accuracy scores from 10-Fold cross check validation, and the average accuracy score of the 10. The ROC Curve plot was dropped because of the layout design balance, but the ROC_AUC score was included. The other consideration in GUI design for this page was after the P-value analysis on each of the selected 17 features for this model, we decided to drop GENDER_FEMALE and combine SURFACE_DRY and SURFACE_WET given the insignificant influence to the model. GUI design was adapted to make this selection feasible.

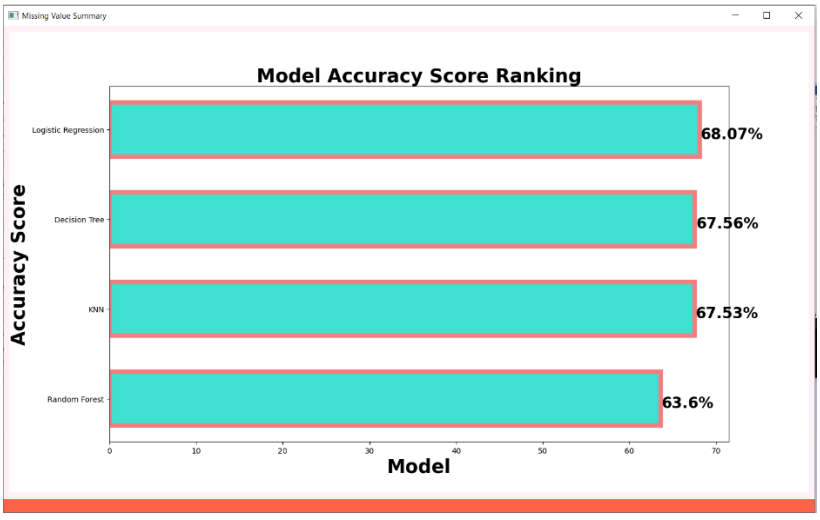



4) **KNN**: QHBoxlayout for the first layer, and QVBOXlayout for both sides as the second layer. Something different and unique of this window compared to others is the "Find the best K" button which will trigger a loop from 1-50 and plot a K value graph to show the best K value.




## F. The "Conclusion" menu

Display the ranking of the accuracy scores from 4 models we built, and the accuracy scores were the average from 10-fold validation. While we didn't evaluate our models only based on the accuracy scores, it is showing anyway from GUI's perspective for completion.




# :: ========================

# :: ========================

## 5. Summary and Conclusions

I really enjoyed this GUI design exercise and felt I've become a basic GUI master at least with basic understanding of how it works and can quick realize any user interaction features through lines in PyQt5.

The major things I've learned are:

- how to customize elements in PyQt5
- how to customize plots using Matplotlib
- how to make user input on windows linked and dynamic with results presented on windows, such as plots and text results

Future improvements and observations:

- Now I understand why few people use Matplotlib as it is much more difficult to draw plots with limited ability to present and to customize
- More work for me to learn how to pass variable values between different windows. Making nested functions and across different classes was one way, but it seems so complicated and can be improved. I will do more research on if there are better ways to do this.

## 6. Percentage of Code Copied

- I learned GUI based on professor's demo from GitHub but it wasn't copy and paste at all, actually none of the lines worked by just copying and pasting and I had to make many modifications. Same as when I take codes from my team mate members, many cases their codes can be used directly in PyQt5 especially for some unique algorithms package and plotting package they used.
- With that background, by my calculation:
- Code lines took directly from team members in my GUI code (excluding data preprocessing and cleaning): 30
- Code lines took directly from internet: 6
    - for posting a picture from the internet to display in a window background
    - for aligning the text on both horizontal and vertical level (you need to use binary operators)
- Total code lines in GUI: 1963
- My own code: 1927 (including space and comments)
- Percentage of code from others: (36-0)/(36+1963)*100 = 1.8%

## 7. Reference


- https://github.com/amir-jafari/Data-Mining/tree/master/Demo/PyQt5
- https://www.tutorialspoint.com/pyqt5/index.htm

- https://zetcode.com/gui/pyqt5/