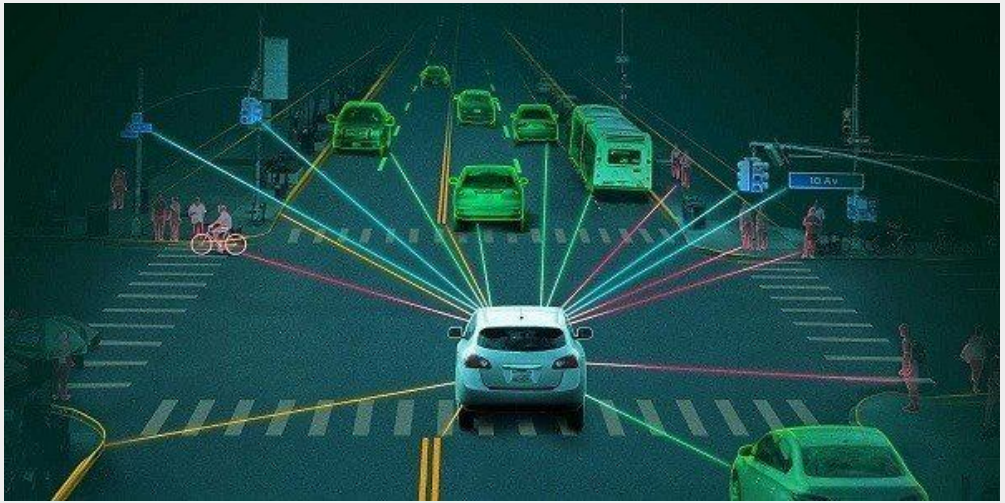
SWITRS: “Separating Driver Classes to Estimate the Safety Benefits of AI-Corrected and AI-Autonomous Vehicles”

Group 5 Project Members

- Micah Simmerman
- Nolan Ollada
- Nathan Palmer

Project Motivation: Identifying High-Risk Driver Classes



Artificial intelligent (AI) driving systems are a new field of automotive research that is based on extending the “awareness” of onboard automotive computing systems with an integrated sensor harness (e.g., radar, lidar, infrared, etc.) to detect current traffic conditions. The information relayed from these sensors can be used to alert the driver to dangerous conditions, or enable the vehicle to make corrective actions independently. These AI safety technologies have enormous potential for reducing the rates of common traffic collisions, preserving the life, health, and safety of vehicle passengers, bicyclists, pedestrians, and drivers other motor vehicles.

Image included courtesy of: https://www.eescorporation.com/do-self-driving-cars-use-ai/

SWITRS California Collision Database (Kaggle)

- SWITRS California Collision Database
  - California Highway Patrol (CHP) using the Statewide Integrated Traffic Records System (SWITRS). Records from January 1 2001 to mid-December 2020 compiled on Kaggle by user Alex Gude
  - https://www.kaggle.com/datasets/alexgude/california-traffic-collision-data-from-switrs
  - SWITRS is a large SQLite database file containing 3 main tables with dozens of attribute columns, and almost 10 million data points.

Follow the directions in the repository documentation to try these tools for yourself!

Data Extraction and Preprocessing

```
# NOTE: always store the switrs.sqlite file in the parent directory where this python notebook is kept.
db_file = '../switrs.sqlite'  # name the db file downloaded directly from https://www.kaggle.com/datasets/alexgude/californi
sqliteConnection = sqlite3.connect(db_file)

# if os.path.isfile(db_file):  # determine if there is any SQLite db file by that name exists in the specified file location.
sqliteConnection = sqlite3.connect(db_file)  # establish a connection if the file does exist.
cursor = sqliteConnection.cursor()  # create a cursor object.
table_query = 'SELECT name from sqlite_master where type= "table"'
cursor.execute(table_query)
result = cursor.fetchall()
```

1. Clone the repository folder to your local computer.
2. Download the zipped archive containing the SWITRS SQLite database file, unzip the folder and place the sqlite file in the parent directory of the .ipynb notebook.
3. SQLite Journaling Mode = ‘OFF’.
4. Drop the uninteresting attribute columns using an SQL Alchemy cursor.

Generating the Random Sample Frames

1. Randomly collect 10,000 samples from the ‘collisions’ table.
2. Extract the ‘case_id’ primary key values from this data frame.
3. Use the extracted case_id’s ist to collect the corresponding data frames from the ‘parties’ and ‘victims’ tables.
4. (OPTIONAL) mutually-exclusive repeat sampling to produce an alternate set of frames.

Cleaning and Examining the Data

Address NULL Collision Values

```
# Start with Numeric Attributes
numeric_collision_columns = collisions_n_10000_training_set_df.select_dtypes(include=['float64', 'int64']).columns
for col in numeric_collision_columns:
    print("Column:", col)
    print(collisions_n_10000_training_set_df[col].value_counts())
    print("\n")
# Most of the numeric columns can have 0 filled in for their NaN values. Lets deal with special cases first.

# PARTY COUNT: The most common and intuitive value for this will be 2.
collisions_n_10000_training_set_df['party_count'] = collisions_n_10000_training_set_df['party_count'].fillna(2)
# For the rest of the columns, it makes most sense to fill NaN with 0.
# Note: filling Latitude and Longitude with 0 doesnt make much logical sense, but for the sake of avoiding errors w
# If using Lat and Long in the future, consider excluding entries with 0 value.
for col in numeric_collision_columns:
    collisions_n_10000_training_set_df[col] = collisions_n_10000_training_set_df[col].fillna(0)
```

1. Observe the dtypes and variable outcomes in each data frame.
2. Separate numeric/convertible numeric and nominal attribute columns.
3. Replace null values, apply smoothing, convert boolean to numeric.
4. Reexamine and repeat.

Numerical Principle Component Analysis (PCA)

```
for i in range(len(collision_parties_numeric_index)):  # shorten to test the algorithm.
# for i in range(4):  # shorten to test the algorithm.  # shortened version for testing.
    for j in range(len(collision_parties_numeric_index)):
    # for j in range(4):  # shortened version for testing.
        if i != j:
            title = "Numerical PCA:" + "'" + collision_parties_numeric_index[i] + "'" + " vs. " + "'" + collision_parties_numeric_index[j] + "'
            plt.title(title)
            plt.scatter(X_pca[:, i], X_pca[:, j])  # change the values at these index definitions to control which variables are compared.
            # plt.xlabel(collision_parties_numeric_index[i] + " (as-is)")  # replace these labels with the column name
            # plt.ylabel(collision_parties_numeric_index[j] + " (as-is)")
            # plt.xlabel(collision_parties_numeric_index[i] + " (min-max normalized)")  # replace these labels with the column name
            # plt.ylabel(collision_parties_numeric_index[j] + " (min-max normalized)")
            plt.xlabel(collision_parties_numeric_index[i] + " (z-score normalized)")  # replace these labels with the column name
            plt.ylabel(collision_parties_numeric_index[j] + " (z-score normalized)")

            # plt.show()  # TODO: replace .show() statement with a repeated write to a specified external pdf file.
            plt.savefig(pdf_file, format='pdf')  # append each figure to the indicated .pdf file.
            print(title, "is complete.")
            print("Figure count so far:", fig_count)
            plt.clf()  # clear the figure each time or the plots get overlaid on every subsequent graph.
            fig_count += 1

pdf_file.close()  # close the pdf file when we are done.
print("See 'df_collision_parties_std_scaler_numerical_pca.pdf' for results!")
```

- $O(n^2)$ algorithmic complexity.
- Generates 2D attribute cross-dimension plots to examine every dimension of the inner join between ‘collisions’ and ‘parties’. An EXHAUSTIVE search.
- Results are very insightful, but tedious to examine. See Numerical_PCA_Report.docx.

Categorical PCA

- Categorical attributes extracted from dataframe
- Chi-squared tests on pairs of attributes to find associations
- Highly associated attributes saved for further examination
- Manually filtered highly associated attributes to further reduce dimensionality

```
Columns: ('location_type', 'collision_severity')
Chi-squared: 31.4739, p-value: 0.0001
------
Columns: ('location_type', 'road_surface')
Chi-squared: 15.3754, p-value: 0.0175
------
Columns: ('location_type', 'road_condition_1')
Chi-squared: 35.6775, p-value: 0.0004
```

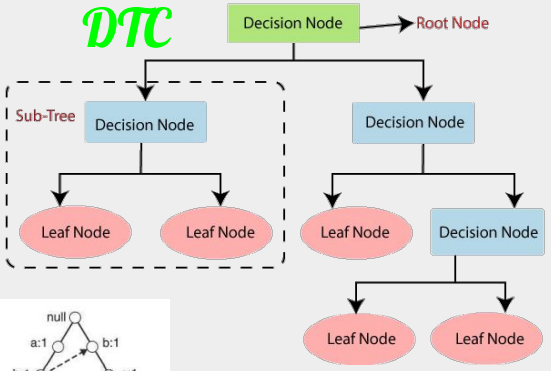
Examining Data Cube Trends

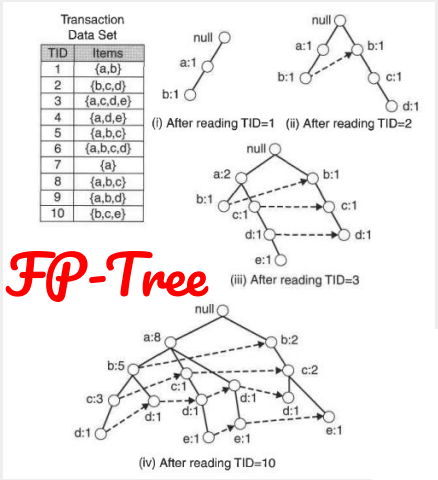
- Data Cube main trends
- Inexperienced drivers are responsible for most accidents.
- Weather conditions (mostly wet / fog) increase accident rates significantly for inexperienced drivers as well as tenured drivers.
- Night collisions in non-urban areas have a much higher fatality rate than night collisions in urban areas.
- Pedestrian accidents occur mostly at night in urban settings, fatality rates are higher at night than during the day.

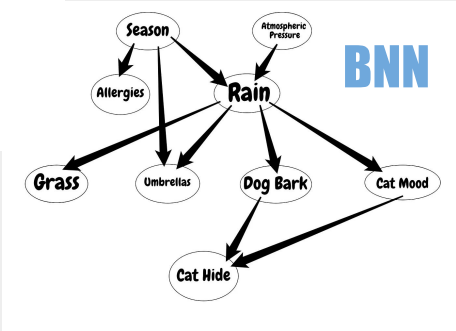
Verifying Trends with Classification Models

- **Decision Tree Classification (DTC)**
- **FP-Tree Based Classification**
- **Bayesian Belief Network (BBN Classification)**

Images provided by: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm, https://blog.knoldus.com/machinex-understanding-fp-tree-construction/, and https://towardsdatascience.com/introduction-to-bayesian-belief-networks-c012e3f59f1b.







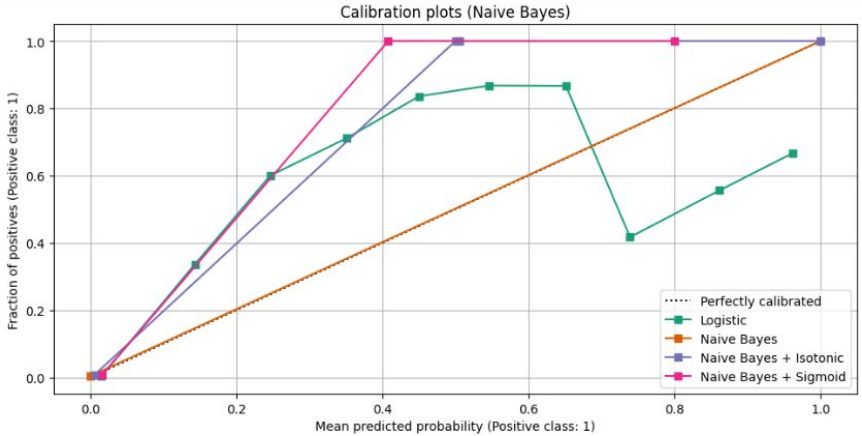
Decision Tree Classifiers and the sklearn Class Rule Generator

```
X_temp = df_collision_parties[['alcohol_involved', 'cellphone_in_use', 'party_age', 'party_count', 'motorcycle_collis
'motorcyclist_killed_count', 'other_visible_injury_count', 'party_number', 'intersection', 'killed_victims', 'party_nu
'injured_victims', 'bicyclist_killed_count', 'pedestrian_collision', 'school_bus_related', 'pedestrian_killed_count',

X = X_temp[~X_temp['alcohol_involved'].isin(['10000 to 25000'])]  # unknown problematic string value encountered in th
# X = std_scaler.fit_transform(X_temp)  # apply standard normalization. (OPTIONAL)


y = df_collision_parties[collision_parties_numeric_index]  # X and y must refer to the same data frame.
# y = std_scaler.fit_transform(y_temp)  # apply a transform to this column. (OPTIONAL)
y = y[['bicycle_collision']].values  # refers to the outcome attribute only. We use 'bicycle_collision' because it is

# note that this library package performs train-test-split automatically. We may wish to increase the size of the data
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.99, random_state=42)  # perform a train-test-split
```



- Python sklearn offers fantastic tutorials with online documentation.
- Examine clustering features and regression trends identified during Numerical PCA.

FP-Trees

- mlxtend library
  - Convert collisions to transaction format
  - Apply FP growth algorithm
- Able to find frequent patterns associated with a given association rule. This allowed us to find frequent patterns in collisions resulting in the death of at least 1 person

Support: 0.06666666666666667, Itemsets: location_type: highway, party_sobriety: had been drinking, under influence

Bayesian Belief Networks (BBN)

- Utilized Bayesian Belief Network model to determine probabilistic relationships between variables and specific outcomes.
- Noted patterns with age's impact on fatal accidents as well as experience.
- Inclement weather increases the rate of fatal accidents for this group.

- Using this method efficiently aggregates and organizes complex data, enabling quick insights into trends and relationships across multiple dimensions.

Key Findings

- Inexperienced drivers are more likely to be involved in a collision of any kind. Young drivers (and their parents) appear to be good candidates to become an automotive AI safety feature market class.
- Solo drivers experience an increased rate of traffic collisions compared to drivers with one passenger. This trend is best observed in the figures of “Numerical_PCA_Report.docx”.
- This “solo trend” appears to suggest that these drivers would benefit from from an “AI-copilot” system to alert the driver whenever they become drowsy, distracted, are ability-impaired, or are attempting to participate in an unsafe driving behavior (such as brushing teeth, shaving, or applying cosmetics).
- Highways are the most common location of fatal accidents.
- Many fatal accidents involve an airbag that failed to deploy.
- Left turns are the most dangerous vehicle maneuver.

Proposed Future Works

- Expand the analysis by integrating data from AI vision systems, autopsy reports, weather monitoring systems, etc.
- Perform a numerical PCA using a data frame composed of an inner join between the ‘collisions’ + ‘victims’ tables.

Visit our Github Repository!

https://github.com/Micah614/Group5_Data_Mining_Project/tree/main

Works Cited

- Mubarak Almuntairi, Kashif Muneer, and AqeelUrRehman. 2022. Vehicles Auto Collision Detection & Avoidance Protocol. IJCSNS International Journal of Computer Science and Network Security, VOL.22 No.3, March 2022. http://paper.ijcsns.org/07_book/202203/20220315.pdf
- Alex Gude. 2021. California Traffic Collision Data from SWITRS. Retrieved from https://www.kaggle.com/datasets/alexgude/california-traffic-collision-data-from-switrs
- Hovannes Kuhandjian. 2022. AI-based Pedestrian Detection and Avoidance at Night Using an IR Camera, Radar, and a Video Camera. CSU Transportation Consortium.Project 2127. https://scholarworks.sjsu.edu/mti_publications/430/
- Hazem H. Refai and Fadi Basma. 2009. Collision Avoidance System at Intersections FINAL REPORT - FHWA-OK-09-06. Electrical and Computer Engineering Department. https://www.odot.org/hqdiv/p-r-div/spr-rip/library/reports/fhwa-ok0906.pdf
- github.com/agude. 2021. SWITRS-to-SQLite. SWITRS data dictionary retrieved from https://github.com/agude/SWITRS-to-SQLite/blob/master/switrs_to_sqlite/value_maps.py.
- kaggle.com/silversurf. 2018. US Traffic - Getting to the Bottom of it. kaggle.com data analysis. Retrieved from https://www.kaggle.com/code/silversurf/us-traffic-getting-to-the-bottom-of-it/notebook.
- Tesla Deaths. 2023. Retrieved from https://www.tesladeaths.com/.
- Brijesh Saluja. 2023. “What are the Ethical and Safety Concerns with Autonomous Vehicles?”. https://community.nasscom.in/communities/digital-transformation/what-are-ethical-and-safety-concerns-autonomous-vehicles.
- August, 2023. “Automatic Artificial Intelligence Market Report”. marketsandmarkets.com; Report Code: SE 5533. https://www.marketsandmarkets.com/Market-Reports/automotive-artificial-intelligence-market-248804391.html
- Analysis of traffic injuries and fatalities in the U.S. since 1966 https://www.kaggle.com/code/silversurf/us-traffic-getting-to-the-bottom-of-it
- Dataset of Tesla deaths including autopilot usage since 2013 https://www.kaggle.com/datasets/thedevastator/tesla-accident-fatalities-analysis-and-statistic?select=Tesla+Deaths+-+Sudden+Acceleration.csv