

Insurance Claims System and Intelligence

Luther Nicholas

Park University

Problem Statement

The management of Milberg Insurance has credited this project to the development and deployment of a model to direct the deployment of resources towards claims with potential fraud. The present system yields cost, time, legal, and service quality inefficiencies that ultimately result in potential faulty business analysis solutions, longer claim life-cycles for customers, and operational cost inefficiencies. This problem directly or indirectly impacts existing consumers, management, and shareholders. Competitively, poor service quality, with regards to fraud investigations, has not only succeeded in the loss of market share, but also alters the product's appeal towards prospective customers. Investing all claims for potential fraud provokes operational inefficiency, and even legal costs, which adversely affect the company's profitability. Notably, potential faulty business intelligence may have triggered defective premium pricing decisions and risk predictive capabilities, thus producing operational and cost inefficiencies.

The primary models will inform the deployment of resources towards fraud investigations by detecting or predicting potential fraudulent claims based on select factors. All these are dependent upon the production of logical analytical procedures, and data-centric, interactive, and empowering visualizations. The data will be sourced from [GitHub](#).

The waterfall method will guide the flow of defined tasks and processes towards the materialization of the project's goal and requirements. Alexander (2020) lists waterfall consecutive phases as "requirements, design, implementation, verification, deployment and maintenance". The method is relevant, as the size of the project is small and project specifications are clearly defined.

Project Schedule

Schedule

Week	Deliverables from List
2	Information Systems Proposal
3	Data Collection and Cleaning
4	Data Exploration
5	Data Analysis & Results
6	Conclusions and Benefits
7	Dashboard Creation for Decision Makers

Information Systems Proposal. The goal of this step is to plan and develop a collection of computer-based tools to help with collection, storage, and processing of data into useful information. The components accessed for design include the role of the hardware, software, user, and data source. The expected outcome is a clear plan that defines system components together with an optimal set-structure of the components for results.

Data Collection and Cleaning. The objective of concluding this goal is to clearly define the variables. It is in this step where the data will be parsed, corrected, standardized, and consolidated to fit analysis goals. The technologies expected to be used in this step are Python and Tableau. The anticipated result is clean and well-defined dataset(s) or dataframe(s) ready for exploration and analysis.

Data Exploration. The goal of this step is to use visual exploration techniques to understand the characteristics of the data in the dataset. To initiate data analysis insight, this step aims at exploring possible relationships among variables in the dataset. The technologies expected to be used in this step are Python and Tableau. The expected outcome is visuals that clearly define the existing relationships in the dataset, helping with analysis, interpretation and reporting.

Data Analysis & Results. The goal of completing this goal is to build and test models, and create visualizations for business analysis. Technologies anticipated to be used are Tableau and Python. The expected results include the creation of a model to classify if a claim is fraudulent or not, model to predict insurance total claim amounts (based on vehicle type, location, age, premium), and visualization to show: Average claim per state, Claims per age group, Number of accidents per car model.

Conclusions and Benefits. The primary objective of this goal is to access the applications of the analysis results towards informing strategic decision making. This other objective is to explore and suggest items or ways in which the analysis or model can be improved. The expected outcome is to show how the analysis can be used to solve the ascertained business problem:

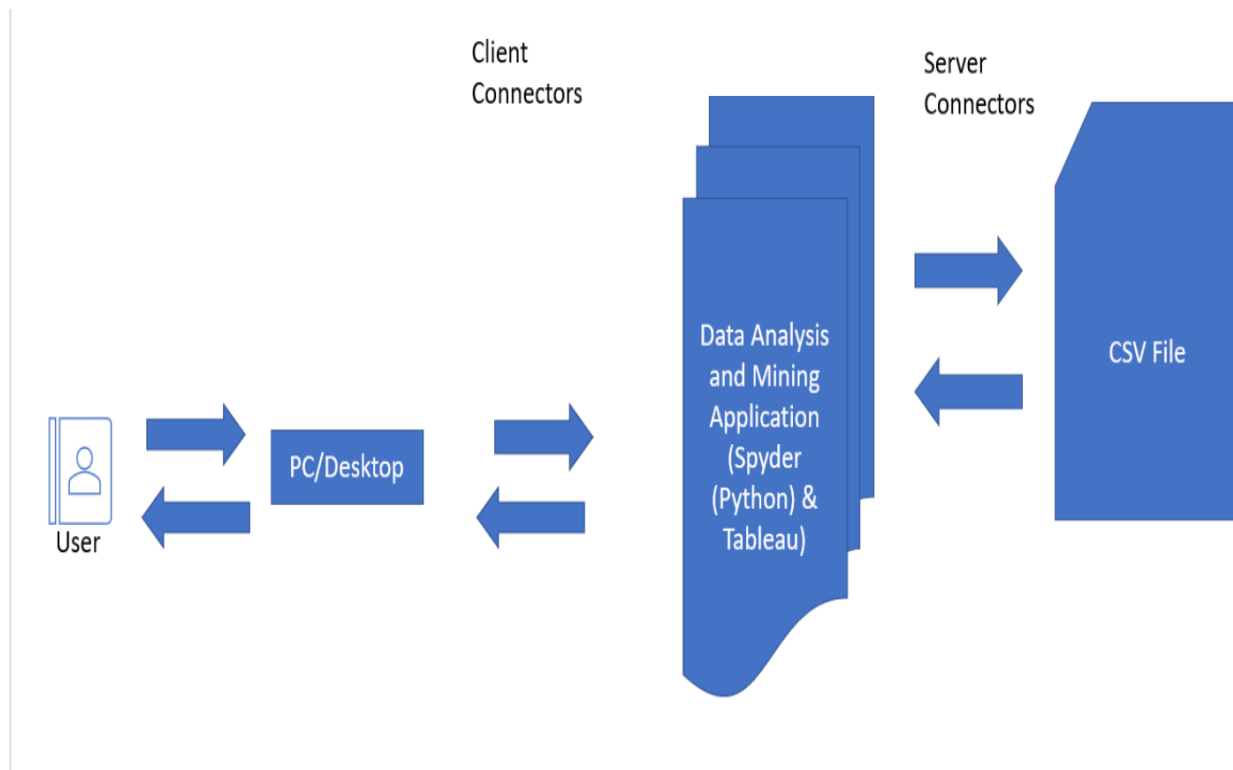
Dashboard Creation for Decision Makers. The goal of this analysis is to create interactive dashboards summarizing the data insights, analysis, results, and suggested business applications. The tools that can be used in this step are Tableau Desktop and Microsoft PowerPoint. The expected outcome are dashboards that display necessary information effectively and clearly to support prompt business decision making.

Information Systems Proposal

System Architecture Option with its Rationale.

A 3-tier client-server architecture model is to support the extraction of business intelligence models, reports, and visualizations. The model is optimal, as it partitions tasks and workloads between the server and the client. The client's portion of the architecture serves the user's requests while servers supply responses to the user. The selected model architecture also supports data independence, multiple analysis, and presentation of information from a primary data file (server) (Dorofeev, & Shestakov, 2018). Essentially, with the adopted system, multiple analysis may be served with a server, while each analysis is independently in connection with only one server. This structure allows multiple analysis to request and receive interactive or static reports and results for solid and data-centric business insights and decision making.

The System Diagram



Desktop/Tablet. Helps the user request, edit, and publish interactive analysis reports from the server. The user can also use the hardware to edit, manipulate, and manipulate the data file through an application such as MS Excel, Tableau, and Spyder (Python). Lastly, the user can also use the hardware to perform and publish statistical and predictive analysis and results, respectively.

Spyder (Python). This is an Integrated Development Environment (IDE) that will extend an environment where the user can generate and use Python codes to edit and manipulate data, perform analysis, and generate reports. Python is an interactive language that will enable the analyst to make, modify, and break codes based on the specific needs of a given analysis or test. Python is ideal not only for its simplicity, but also for its portability feature. The user can easily document and share analysis and reports generation codes with other users in the organization.

Tableau Desktop. This software allows the user to connect to data and analyse it. This includes some data cleansing, data analysis and exploration. Clean data is analysed with Tableau's simple to understand and innovative visual system. The tool also allows an analyst to build interactive report/dashboards to share with organizational users for decision making.

Presentation Applications. This includes software like Microsoft Excel and Microsoft PowerPoint. With these applications, the user can publish, document, and present analysis results and dashboards. The reports generated can be static or interactive. Interactive reports will enable decision makers to interact with data results in considerable detail (could be through filtering or sorting).

Dataset

The dataset used for analysis is titled "insurance_claims set" and it is sourced from GitHub. It is a sample of 1000 insurance claim cases collected from seven states. The dataset details aspects about a given incident/accident, policy details, insured details, and claim details.

Link: https://github.com/mwitiderrick/insurancedata/blob/master/insurance_claims.csv

Data Cleaning

Dropping Irrelevant Columns, & Dropping or Interpolating Missing Data

In Microsoft Excel, the find-and-replace feature is used to replace the "?" value with null in columns, "property_damage" and "police_report_available". The successive dataset is then loaded in Python as a data frame using pandas. The preprocessing package from the sklearn library is imported as pp. The package supports data cleaning in context, as it aids the transformation of relevant categorical values into numerical values.

```
import pandas as pd  
from sklearn import preprocessing as pp  
  
df = pd.read_csv('C:/Users/18166/OneDrive/Documents/project.csv')
```

`df.info()`

#	Column	Non-Null	Count	Dtype
0	months_as_customer	1000	non-null	int64
1	age	1000	non-null	int64
2	policy_number	1000	non-null	int64
3	policy_bind_date	1000	non-null	object
4	policy_state	1000	non-null	object
5	policy_csl	1000	non-null	object
6	policy_deductable	1000	non-null	int64
7	policy_annual_premium	1000	non-null	float64
8	umbrella_limit	1000	non-null	int64
9	insured_zip	1000	non-null	int64
10	insured_sex	1000	non-null	object
11	insured_education_level	1000	non-null	object
12	insured_occupation	1000	non-null	object
13	insured_hobbies	1000	non-null	object
14	insured_relationship	1000	non-null	object
15	capital_gains	1000	non-null	int64
16	capital_loss	1000	non-null	int64
17	incident_date	1000	non-null	object
18	incident_type	1000	non-null	object
19	collision_type	1000	non-null	object
20	incident_severity	1000	non-null	object
21	authorities_contacted	1000	non-null	object
22	incident_state	1000	non-null	object
23	incident_city	1000	non-null	object
24	incident_location	1000	non-null	object
25	incident_hour_of_the_day	1000	non-null	int64
26	number_of_vehicles_involved	1000	non-null	int64
27	property_damage	640	non-null	object
28	bodily_injuries	1000	non-null	int64
29	witnesses	1000	non-null	int64
30	police_report_available	657	non-null	object
31	total_claim_amount	1000	non-null	int64
32	injury_claim	1000	non-null	int64
33	property_claim	1000	non-null	int64
34	vehicle_claim	1000	non-null	int64
35	auto_make	1000	non-null	object
36	auto_model	1000	non-null	object
37	auto_year	1000	non-null	int64
38	fraud_reported	1000	non-null	object

types: float64(1), int64(17), object(21)
memory usage: 304.8+ KB

Fig. 1. Summary of the primary DataFrame

The code below drops irrelevant columns for exploration and analysis. Columns, 'property_damage', 'police_report_available', are dropped because they hold large amounts of data missing values as shown in Fig. 1. Data interpolation, a statistical-based data preprocessing method to estimate missing values based on precedent values, is unfeasible due to the certainty that the aforementioned columns don't hold sets of discrete variables. The subsequent data frame doesn't bear any rows with missing values. A new csv file titled "exploredataset" is derived for data exploration and analysis.

```
df = df.drop(['policy_number', 'policy_bind_date', 'policy_state', 'policy_csl', 'insured_zip',
'insured_occupation', 'insured_hobbies', 'insured_hobbies', 'insured_relationship', 'capital_gains', 'capital_loss', 'incident_date', 'collision_type', 'incident_city', 'incident_location',
'injury_claim', 'property_claim', 'vehicle_claim', 'auto_model', 'umbrella_limit',
'property_damage', 'police_report_available'], axis=1)
```

```
df.to_csv('exploredataset.csv')
```


Categorical Values Conversion into Numerical/Data Labelling

The train and test linear regression and classification models, all categorical variables ought to be converted with numeric values. These columns are 'incident_type', 'insured_education_level', 'insured_sex', 'incident_severity', 'authorities_contacted', 'incident_state', and 'auto_make'. The label encoder in the preprocessing package will automatically assign numeric values for each member of a given column containing categorical values. The steps are explained in the first set of code that encodes the categorical values in the 'incident_types' column in numeric and designates those numeric values into a new column, 'incident_types_encoded'. The numerical code for each categorical variable in a given column are detailed in Fig. 2. A new csv file titled “analysisdataset” is derived for data exploration and analysis.

```
coder = pp.LabelEncoder() #creation of the encoder
array = df[['incident_type']].values #extraction of the column ('incident_type') for encoding
coder.fit(array.ravel()) #change the new returned column into a one dimensional array
array_encoded = coder.transform(array) #transforming the categorical values into numeric
(encoding)
df['incident_type_encoded'] = array_encoded #creation of a new column with encoded values
for the given column with categorical values.
```

```
coder = pp.LabelEncoder()
array = df[['insured_education_level']].values
coder.fit(array.ravel())
array_encoded = coder.transform(array)
df['insured_education_level_encoded'] = array_encoded
```

```
coder = pp.LabelEncoder()
array = df[['insured_sex']].values
coder.fit(array.ravel())
array_encoded = coder.transform(array)
df['insured_sex_encoded'] = array_encoded
```

```
coder = pp.LabelEncoder()
array = df[['incident_severity']].values
coder.fit(array.ravel())
array_encoded = coder.transform(array)
```

```
df['incident_severity_encoded'] = array_encoded
```

```
coder = pp.LabelEncoder()
array = df[['authorities_contacted']].values
coder.fit(array.ravel())
array_encoded = coder.transform(array)
df['authorities_contacted_encoded'] = array_encoded
```

```
coder = pp.LabelEncoder()
array = df[['incident_state']].values
coder.fit(array.ravel())
array_encoded = coder.transform(array)
df['incident_state_encoded'] = array_encoded
```

```
coder = pp.LabelEncoder()
array = df[['auto_make']].values
coder.fit(array.ravel())
array_encoded = coder.transform(array)
df['auto_make_encoded'] = array_encoded
```

```
#df.to_csv('analysisdataset.csv')
```

Insured_sex	Insured_sex_encoder #	Incident_state	Incident_state_encoder #
FEMALE	0	NC	0
MALE	1	NY	1
Insured_education_level	Insured_education_level_encoder #	OH	2
Associate	0	PA	3
College	1	SC	4
High School	2	VA	5
JD	3	WV	6
Masters	5	Auto_make	Auto_make_encoder #
MD	4	Accura	0
PhD	6	Audi	1
Incident_type	Incident_type_encoder #	BWM	2
Multi-vehicle Collision	0	Chevrolet	3
Parked Car	1	Dodge	4
Single Vehicle Collision	2	Ford	5
Vehicle Theft	3	Honda	6
Incident_severity	Incident_severity_encoder #	Jeep	7
Major Damage	0	Mercedes	8
Minor Damage	1	Nissan	9
Total Loss	2	Saab	10
Trivial Damage	3	Subaru	11
Authorities_contacted	Authorities_contacted_encoder #	Toyota	12
Ambulance	0	Volkswagen	13
Fire	1		
None	2		
Other	3		
Police	4		

Fig. 2 : Encoded values for each categorical value in a given column.

Data Exploration

The goals of this step are to explore and understand variables independently, and the existing relationships between variables for the enhancement of business intelligence ends of the project. The target variables for exploration and analysis are the number of claims (count), total claim cost, and fraud status. To resume data exploration, the clean dataset is loaded into Tableau Desktop to extract the exploration visuals.

The Number of Claims

Fig. 3 illustrates that the number of accidents per auto make range between 55 and 80, with Subaru being the brand with the most accidents claims and Honda with the least. In effect, there is no significant difference between brands with regards to the number of claims reported. However, there is a significant difference in the number of claims reported between respective states. New York, South Carolina, and West Virginia had roughly twice the number of claims compared to Ohio, Philadelphia, and North Carolina.

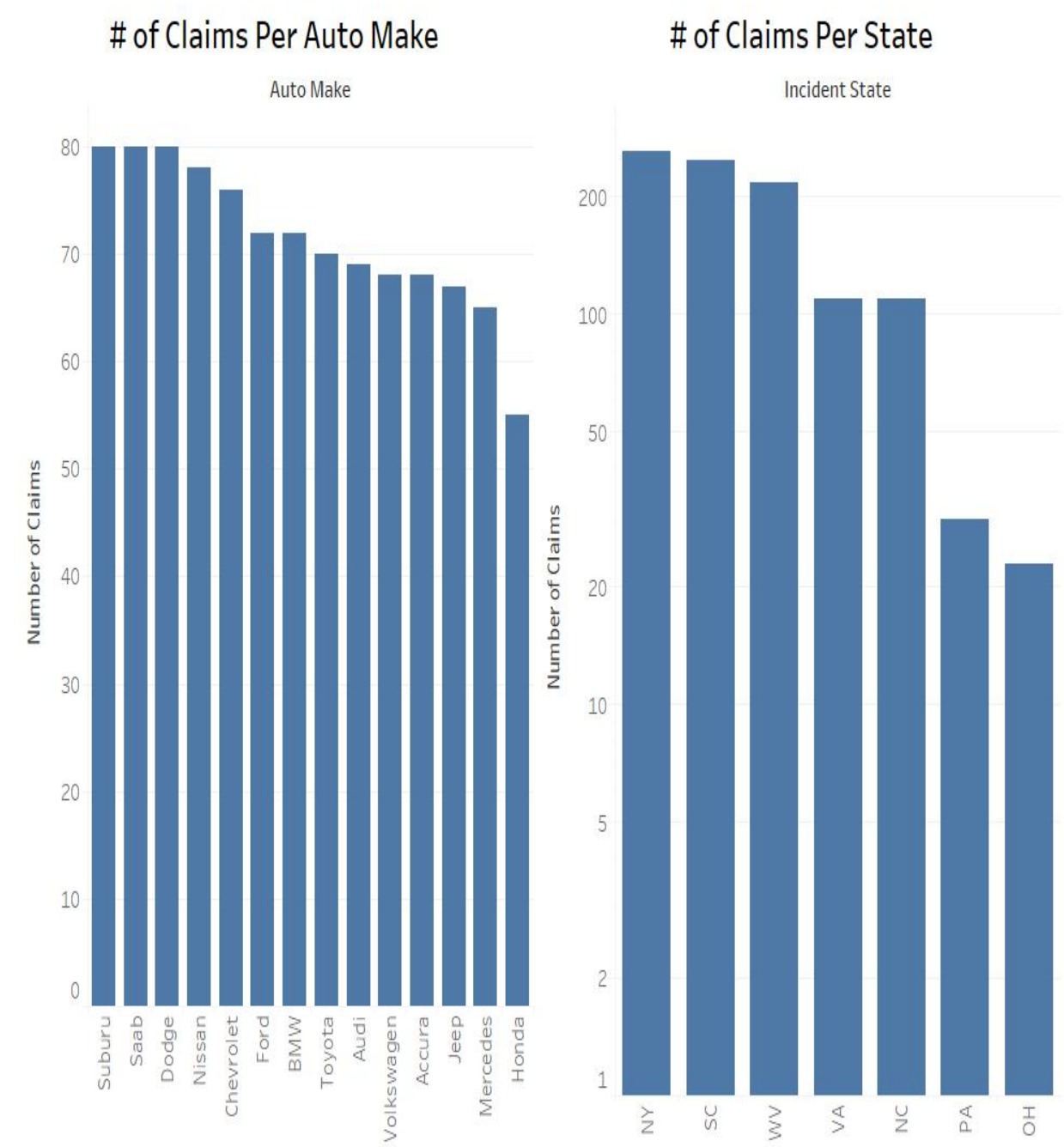


Fig. 3

Total Claim Cost

Total claim cost refers to the total cost incurred by Milberg Insurance in administering and processing individual insurance claims. As shown in Fig. 5 and Fig. 4, there isn't a significant difference in the average total claim cost from one brand to the other. The average claim per brand is around \$50,000. Fig. 5 also illustrates that Volkswagen, Mercedes, and Nissan brands accounted for at least an injury per claim compared to other brands.

Fig. 6 shows a significant higher average claim cost of around \$10,000 for fraudulent claims compared to legit claims. As expected, claims with major damages or total loss incidents had higher average total claim costs compared to claims with minor or trivial damages. Lastly, insured's education level had no significant influence on the average claim cost per claim.

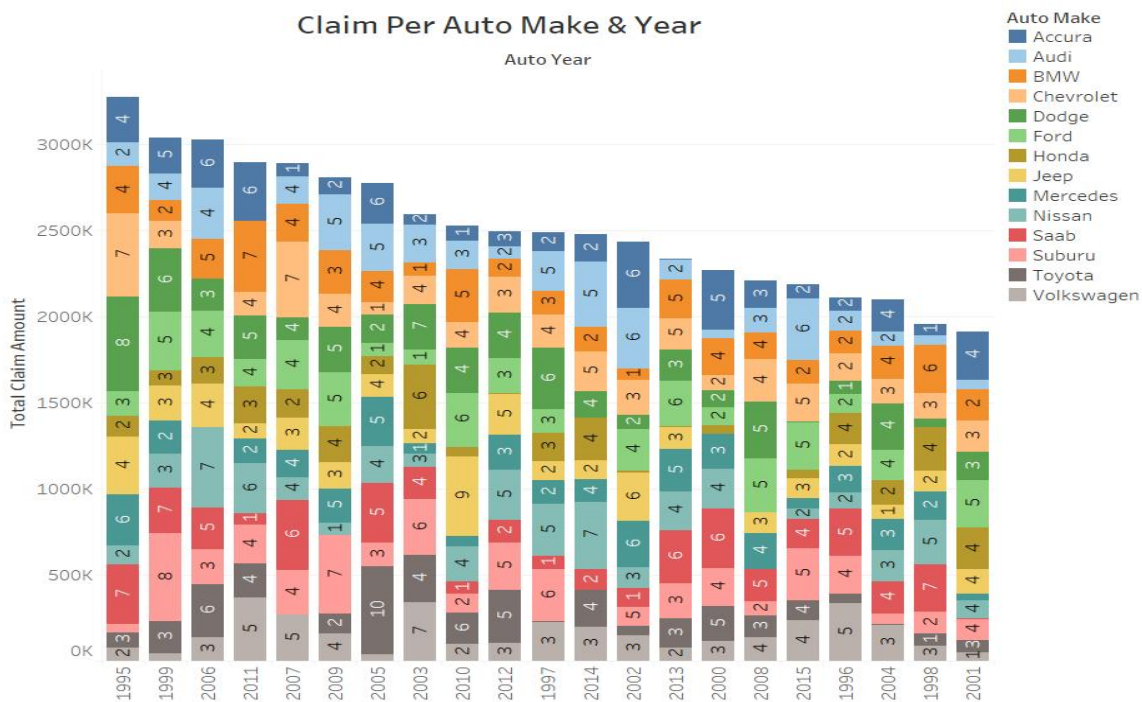


Fig. 4

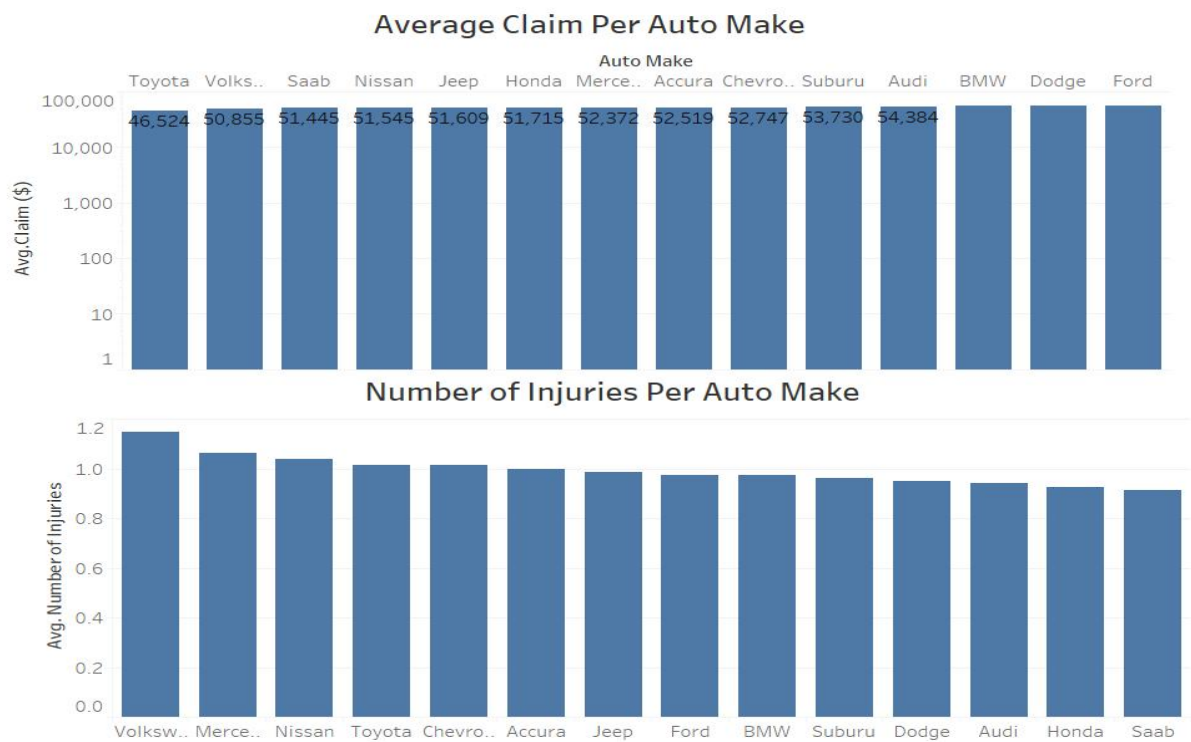


Fig. 5

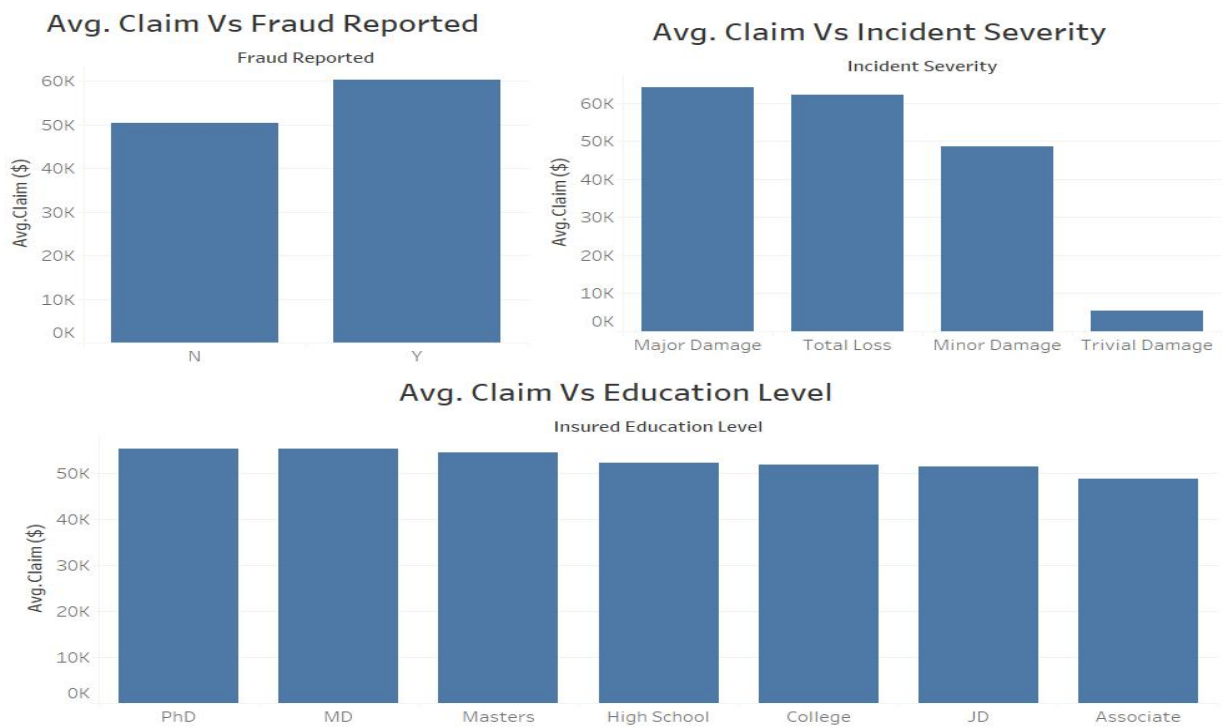


Fig. 6

Fraud Status

The figure below illustrates that fraudulent claims had relatively more major damages incidents reported, while legitimate claims had relatively more minor damages and total loss incidents reported. Most fraudulent claims are accounted for in states, South Carolina and New York. Notably and as expected, in most legitimate claims, police or first responders were contacted, as shown in Fig. 9. Lastly, the insured's education level and incident type did not significantly influence the fraud status of a claim.

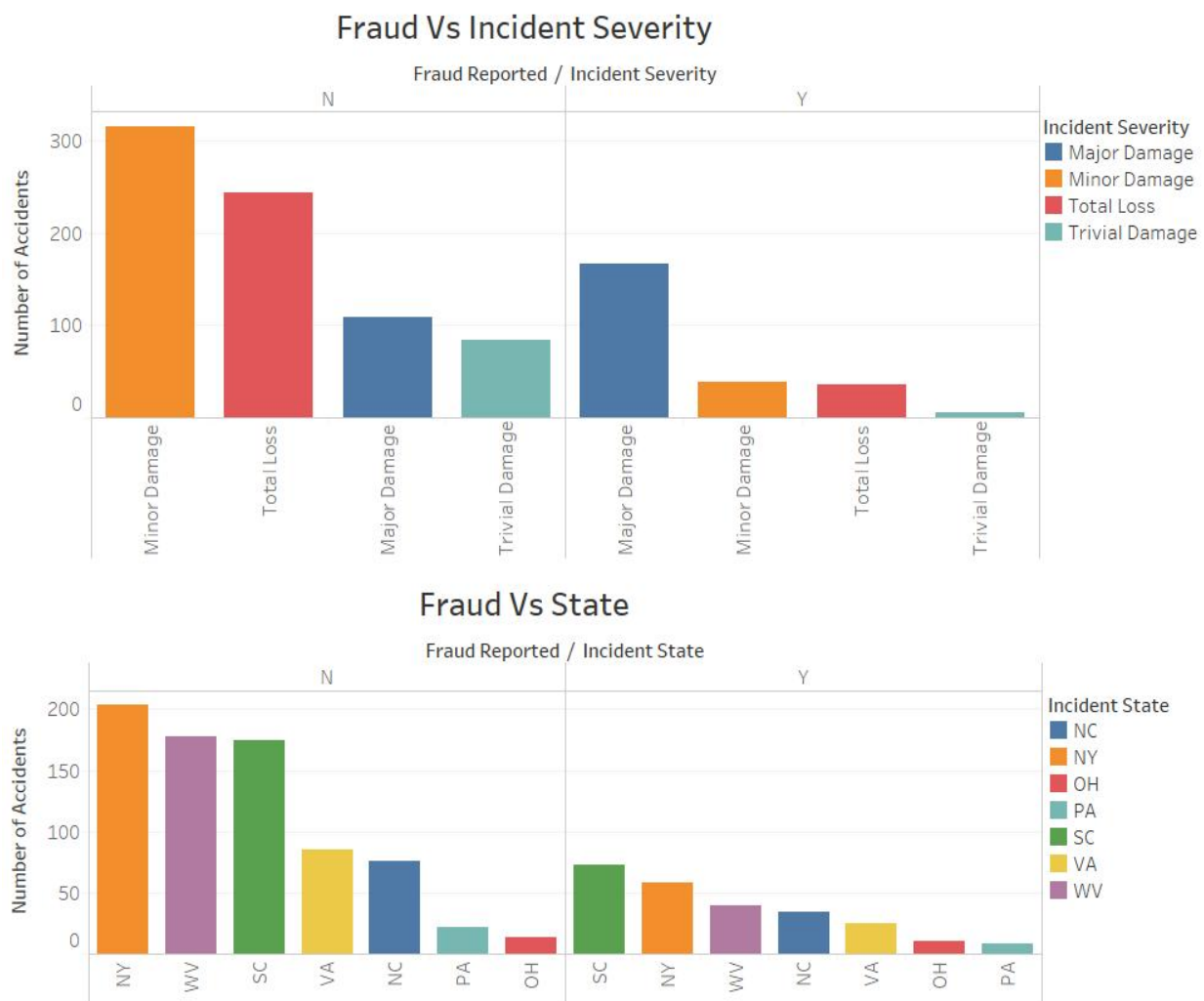


Fig. 7

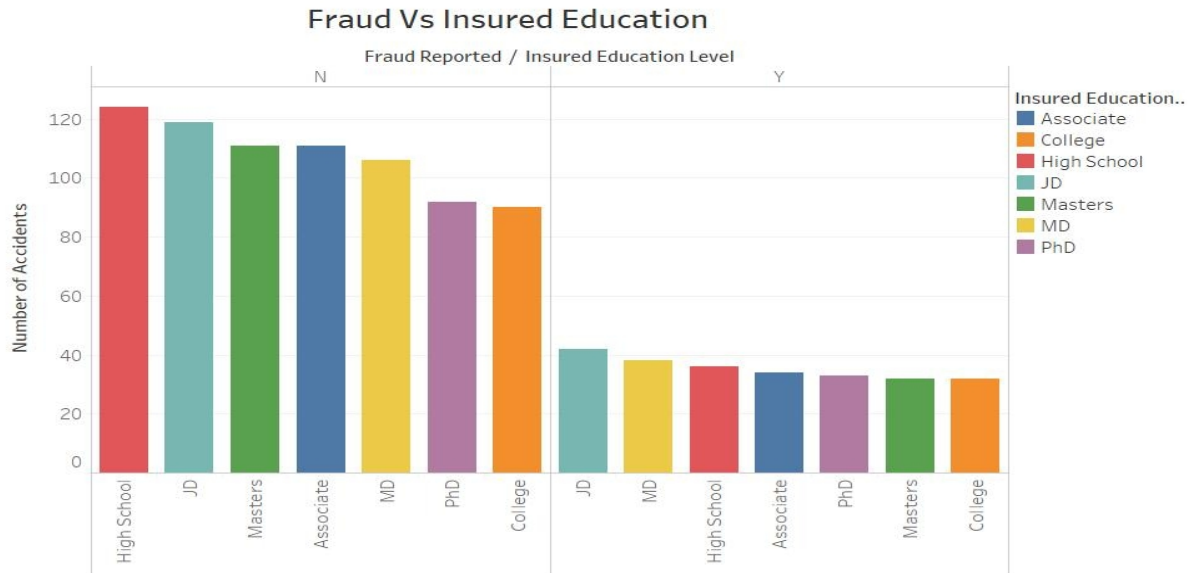


Fig. 8

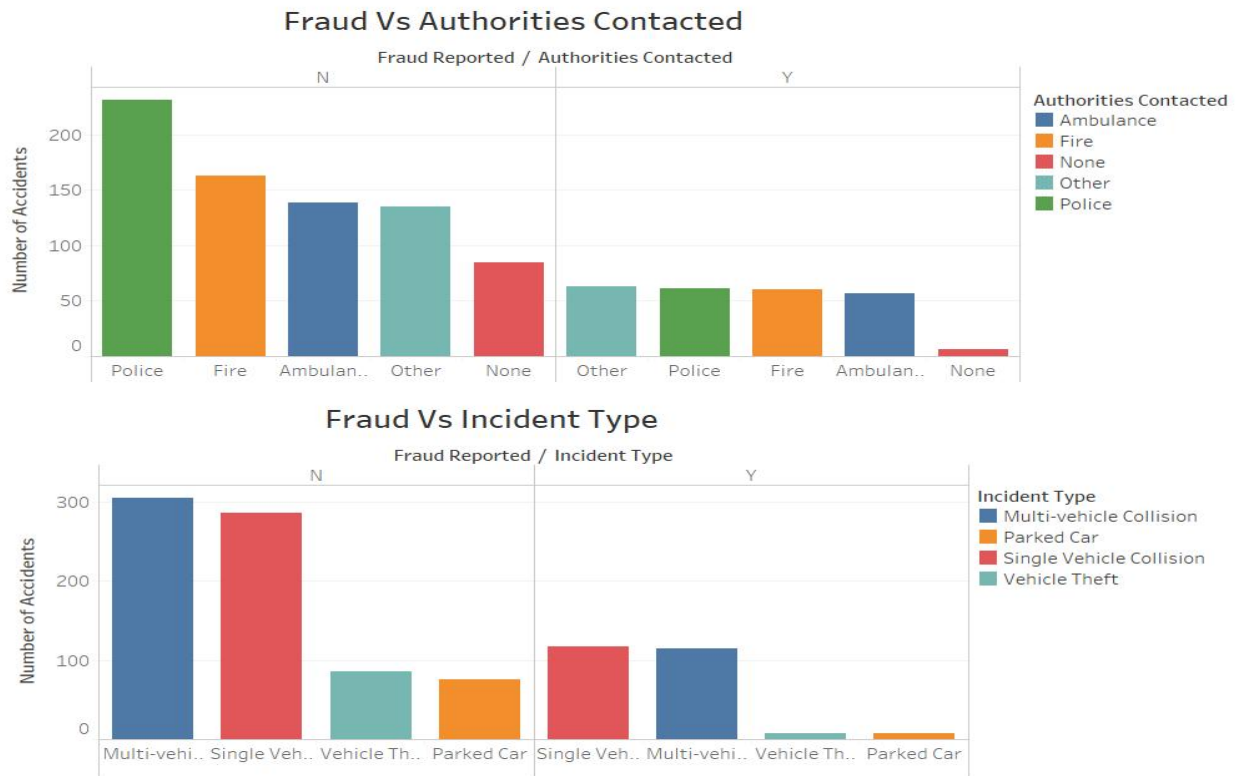


Fig. 9

Data Analysis

Analysis 1. Train, Test, and Score a Fraud Prediction Model

The analysis goal is to create a model to predict if a case is fraudulent or not. The cleaned dataset, intended for analysis, is going to act as the basis of this analysis. The target variable for this analysis is the fraud detailed in the `fraud_reported` column which still holds categorical values (Y and N). As part of data preprocessing, the code below extracts dummy variables from the `fraud_reported` column and assigns them to a new column, `Fraud`. In the new column, fraudulent cases have the value of 1, and genuine case have the value of 0.

```
colms_dummy=['fraud_reported']
ValueDummy = pd.get_dummies(df, columns=colms_dummy)
df = pd.concat((df, ValueDummy), axis=1)
df = df.drop(['fraud_reported', 'auto_make', 'fraud_reported_N', 'incident_state',
'insured_sex', 'insured_education_level', 'incident_type', 'incident_severity',
'authorities_contacted'], axis=1)
df = df.rename(columns = {'fraud_reported_Y' : 'Fraud'})
# NOTE: 1 - Fraud & 0 - No Fraud
df = df.loc[:,~df.columns.duplicated()]
```

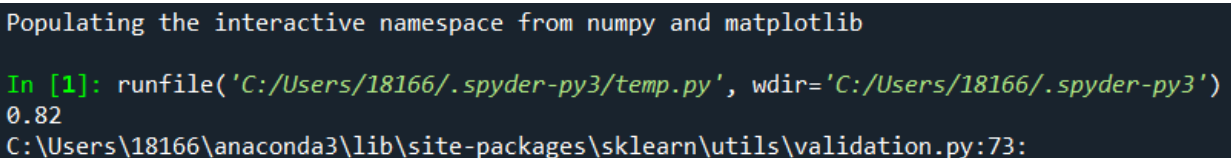
The code below extracts the X and Y value arrays to be use for training and testing the model. The x array values are from all columns, except the new “Fraud” column. Based on the latter analytical objective, the y attribute holds values from the “Fraud” column. The x and y set of values are split into 80%-to-20% chunks. The 80% chunk is used to train the regression model, while the 20% chunk is used to test and score the model. A decision tree classifier object is created and fit with the training set of x and y values for model training. Eventually, the model is tested and scored using the testing set of x and y values for performance.

```
#Extract x and y variables
x = df.drop('Fraud', axis=1)
y = df[['Fraud']]
```

```
#Splitting the dataset into 80% Training and 20% Test
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=1)

#Build and Train Decision Tree Classifier Model
dt_clf = tree.DecisionTreeClassifier(max_depth=4)
#train the model
dt_clf.fit(x_train, y_train)

#Prediction Score
score = dt_clf.score(x_test, y_test)
print(score)
```



```
Populating the interactive namespace from numpy and matplotlib

In [1]: runfile('C:/Users/18166/.spyder-py3/temp.py', wdir='C:/Users/18166/.spyder-py3')
0.82
C:\Users\18166\anaconda3\lib\site-packages\sklearn\utils\validation.py:73:
```

Fig. Analysis 1 result (Prediction Score)

The accuracy of the model to predict if an insurance claim is fraudulent or not is 83%. This score is based on the mean accuracy of the given test labels and data. With an 83% score, the decision tree classifier appears to be a solid predictor of if a claim is fraudulent or not.

The code below presents the x and y values used for training and testing the model into excel dataframes. It also extracts the two Excel dataframes, one with x test and y test values, and the other with y predicted values. These Excel dataframes are then manual combined for comparison (where the prediction is confirmed). Using Excel, the prediction score 83% was the same as the one shown in Python.

```
x_train_data = pd.DataFrame(data=x_train)
y_train_data = pd.DataFrame(data=y_train)
trainingdata = pd.concat((x_train_data, y_train_data), axis=1)
trainingdata.to_excel("traindatavaluesCLASSIFIER.xlsx")

x_test_data = pd.DataFrame(data=x_test)
y_test_data = pd.DataFrame(data=y_test)
testdata = pd.concat((x_test_data, y_test_data), axis=1)
testdata.to_excel("testdatavaluesCLASSIFIER.xlsx")
```

```
predicted_y_values = pd.DataFrame(dt_clf.predict(x_test))
predicted_y_values.to_excel("predictedCLASS.xlsx")
```

Fraud	Predictions	Prediction Status	Prediction Score	
0	0	correct	83%	
0	0	correct		
0	0	correct		
1	0	wrong		
0	0	correct		
1	1	correct		
1	1	correct		

Legend
1 = Fraud Present
0 = Fraud not Present

Fig. Extract of Excel file that shows the model prediction accuracy (individually and collectively).

Analysis 2. Train, Test, and Score a Total Claim Amount Prediction Model

The preceding step, data exploration, guides the analysis as it hints on independent variables which significantly influence the dependent variable. Eventually, using regression analysis, a regression model to predict total claim amounts based on select independent variables will be derived and tested.

Using the code below, the sex variable is excluded from the model as it will institute gender bias which might bring about legal and brand-reputation issues for the business. Some variables like auto year and insured education level are excluded as they displayed insignificant correlation with the dependent variable. It is arguably irrational to include variables like insured hobbies and policy number in the model.

```
#drop columns
df = df.drop(['policy_number', 'policy_bind_date', 'insured_sex', 'auto_year',
'insured_education_level', 'incident_state', 'policy_state', 'policy_csl', 'insured_zip',
'insured_occupation', 'insured_hobbies', 'insured_hobbies', 'insured_relationship', 'capital-
gains', 'capital-loss', 'incident_date', 'collision_type', 'incident_city', 'incident_location',
'injury_claim', 'property_claim', 'vehicle_claim', 'auto_model', 'umbrella_limit',
'property_damage', 'police_report_available'], axis=1)
```

Applying the code below, dummy variables are extracted from columns containing categorical values like incident type and incident severity. For example, from the fraud_reported column in the primary dataset, columns, fraud_reported_Y and fraud_reported_N, are extracted and assigned with dummy variables 1 and 0 (1 indicates presence of a category and 0 indicates absence). To avoid multicollinearity subsequent from dummy traps, one dummy variable from each of the categorical variables is omitted using drop_first=True.

```
#convert categorical values into numerical using dummy values (data preprocessing)
colms_dummy= ['incident_type', 'incident_severity', 'authorities_contacted', 'auto_make',
'fraud_reported']
ValueDummy = pd.get_dummies(df, columns=colms_dummy, drop_first=True)
df = pd.concat((df, ValueDummy), axis=1)
df = df.loc[:,~df.columns.duplicated()]
df = df.drop(['incident_type', 'incident_severity', 'authorities_contacted', 'auto_make',
'fraud_reported'], axis=1)
```

The dependent variables (y) is the total claim amount. The independent variables (x) are months as customer, customer's age, policy deductible, policy annual premium, incident hour, number of vehicles involved, injuries and witnesses, incident type and severity, authorities contracted, auto make, and fraud status.

```
#Extract x and y variables
x = df.drop('total_claim_amount', axis=1)
y = df[['total_claim_amount']]
```

The x and y set of values are split into 80%-to-20% chunks. The 80% chunk is used to train the regression model, while the 20% chunk is used to test and score the model. A linear regression object is created and fit with the training set of x and y values for model training. The coefficient of determination and prediction score (using the testing set of x and y values) are extracted.

```
#splitting data into training and test dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
```

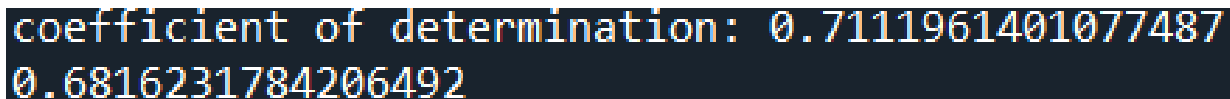
```

linreg = LinearRegression()
a = linreg.fit(x_train, y_train)

r_sq = a.score(x_train, y_train)
print('coefficient of determination:', r_sq)

score = linreg.score(x_test, y_test)
print(score)

```



```

coefficient of determination: 0.7111961401077487
0.6816231784206492

```

Fig. Analysis 2 results (Prediction Score & Coefficient of determination)

The results of the analysis are a coefficient of determination of 71.11%. The model training score is 68.16%. From the R square score, the input variables explain about 71.11% of the variation in the dependent variable. With a p-value less than the 0.5 level of significance, the model to predict total claim amounts is significant. The accuracy of the model to predict total claim amounts using the test x and y set of values is 68.16%.

The code below extracts the x and y values used for training and testing the model into Excel dataframes. It also extracts the two Excel dataframes, one with x test and y test values, and the other with y predicted values. These Excel dataframes are then manually combined for comparison. A chart showing the comparison between the actual and predicted values is derived. An observation of the chart prescribes the model isn't a strong predictor of total claim amounts.

```

x_train_data = pd.DataFrame(data=x_train)
y_train_data = pd.DataFrame(data=y_train)
trainingdata = pd.concat((x_train_data, y_train_data), axis=1)
trainingdata.to_excel("traindatavaluesREGRESSION.xlsx")

x_test_data = pd.DataFrame(data=x_test)
y_test_data = pd.DataFrame(data=y_test)
testdata = pd.concat((x_test_data, y_test_data), axis=1)
testdata.to_excel("testdatavaluesREGRESSION.xlsx")

```

```
predicted_y_values = pd.DataFrame(linreg.predict(x_test))  
predicted_y_values.to_excel("predictedREG.xlsx")
```

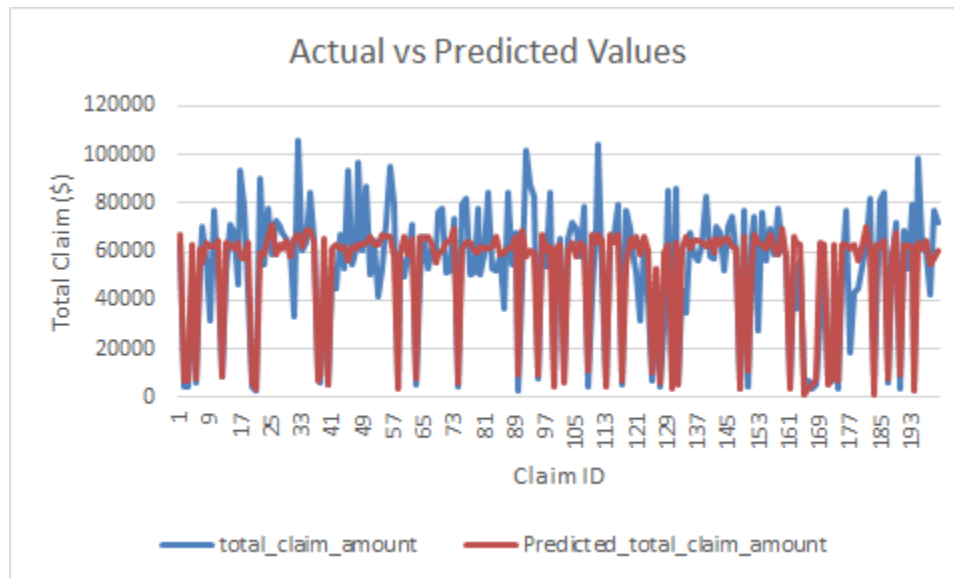


Fig. Actual vs Predicted Values

Interactive Dashboard

Management should consider utilizing the interactive dashboard to access information from dynamic data. Such information will act as a backbone of strategic decision making for the firm. Well-informed and timely decisions will enhance the firm's competitive advantage through strategic formulation and monitoring (Fuchs, 2006).

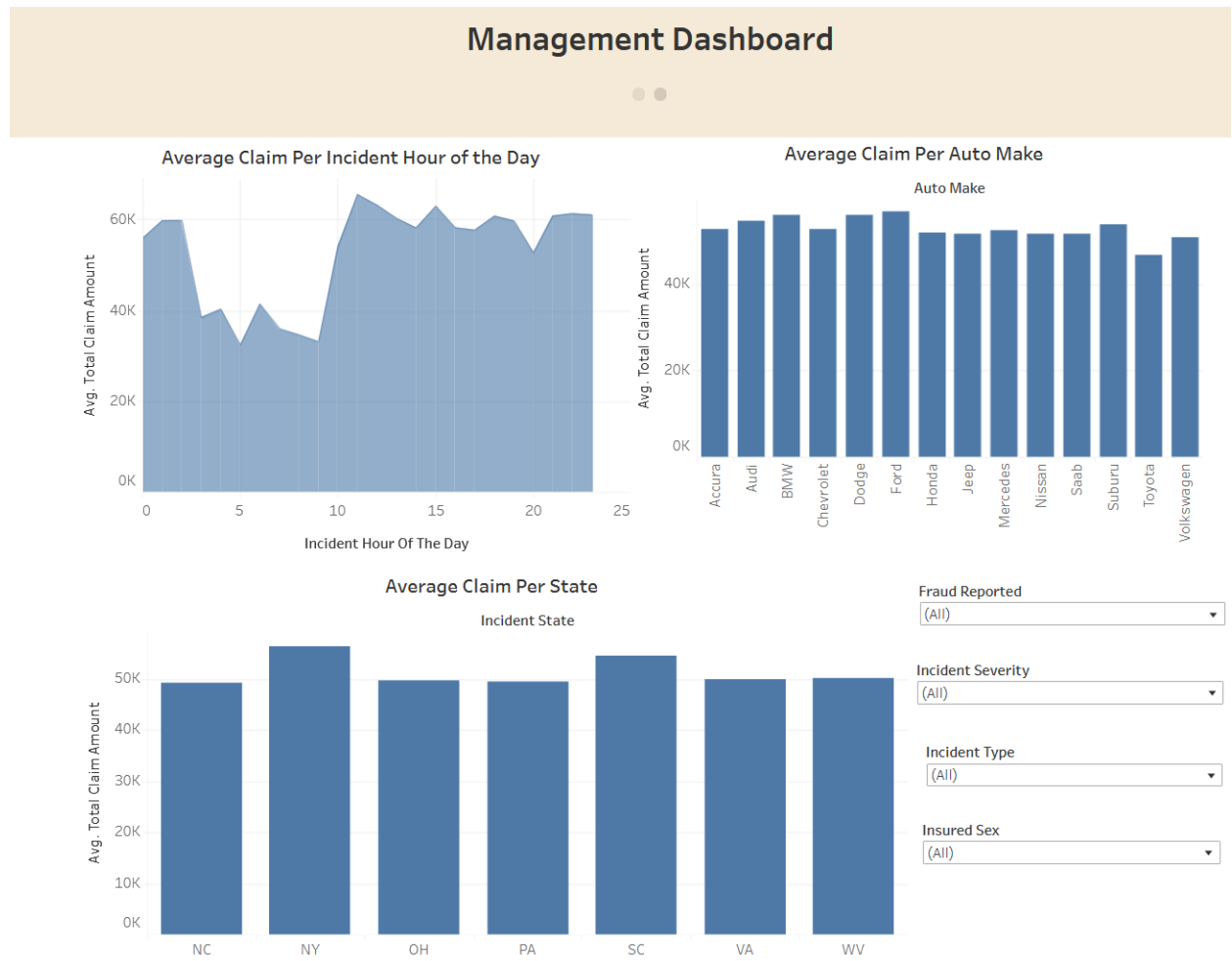


Fig. Interactive Visualization

Conclusions and Benefits

The primary objective of this step is to access the applications of the analysis results towards informing strategic decision making. This other objective is to explore and suggest items or ways in which the analysis or model can be improved. The expected outcome is to show how the analysis can be used to solve the ascertained business problem.

Analysis 1

The firm should use the model that predicts the probability of potential fraud in each claim to determine whether to deploy resources to investigate the claim or not. It's important to note that a fraudulent call by the model doesn't mean a given claim is fraudulent. The outcome of all investigated claims that got flagged by the model for potential fraud must be documented and feed into the classification model. The model will, as expected, improve its prediction performance with more data. Additionally, the model modified and retrained to accommodate market and business requirement dynamics.

The model will benefit the business with resource optimization with respect to insurance claim investigations. This means valuable resources like labor and capital can be redeployed to other business operations, than maximize value and minimize wastage, like customer services and sales. Collectively, all these aforementioned strategies will reduce operational inefficiency, maximize stakeholders' value, and reinforce profitability.

Analysis 2

The firm explores ways to improve the model that predicts the total claim amounts based on select variables like the insured age, insured state, and insured education level. With a 68% prediction score, the model isn't a satisfactory model for guiding premium policy strategies for the firm. If applied (model), the company could risk losing exciting customers to competitors and repulsing potential customers, disrupting marketing strategies, service quality, and competitive advantage.

To improve the model, the firm must explore ways to add more data to the model over time. The data exploration process didn't reveal any outliers that could have altered the prediction score of the model. If improved to yield a satisfactory prediction score, the model will

partially benefit the business with competitive premium pricing strategies. The latter benefit is largely dependent on an external assessment of competitors' strategies and performance.

References

- Alexander, M. (2020). Agile vs. waterfall: Project methodologies compared. *Cio*. Retrieved from <https://www.proquest.com/trade-journals/agile-vs-waterfall-project-methodologies-compared/docview/2448481960/se-2?accountid=28370>
- Dorofeev, D., & Shestakov, S. (2018, November). 2-tier vs. 3-tier Architectures for Data Processing Software. In Proceedings of the 3rd International Conference on Applications in Information Technology (pp. 63-68). Retrieved from <https://dl.acm.org/doi/abs/10.1145/3274856.3274869>
- Fuchs, G. (2006). Beyond the dashboard: Making better decisions. *Business Intelligence Journal*, 11, 30-34. Retrieved from <https://www.proquest.com/magazines/beyond-dashboard-making-better-decisions/docview/222636894/se-2?accountid=28370>