Comprehensive Analysis and Predictive Modeling for Insurance Complaints

College of Professional Studies, Northeastern University

ALY 6140: Python for Analytics Systems Technology

Professor. Daya Rudhramoorthi

March 29, 2024

Rohith Krishnamurthy

(002290948)

Contents

Introduction

The main goal of my project is to build a model that could predict if a complaint filed against an insurance company will be confirmed or not. The model uses past data on complaints to identify patterns and factors that lead to confirmed complaints. By understanding these patterns, insurance companies can take action to fix the issues before they happen and improve their customer service.

To achieve this goal, I will first explore and analyze the complaint data to understand its key features and characteristics. This is called Exploratory Data Analysis (EDA). During EDA, I will create visuals like charts and graphs to examine things like the most common reasons for complaints, how complaint numbers vary across insurance companies, and how long it typically takes to resolve complaints.

After the exploratory analysis, I will use predictive modeling techniques to build the model for forecasting whether future complaints will be confirmed or not. The techniques I employ directly align with and help answer the key question - **what factors influence if a complaint gets confirmed?** By identifying these influencing factors from the data, the predictive models can then forecast complaint confirmation for new cases.

Data Extraction

The dataset used in this project is sourced from(https://data.texas.gov/resource/ubdr-4uff.csv) a publicly available repository containing insurance complaints filed against various companies comprising **248K rows and 17 columns**. The dataset comprises details such as complaint numbers, companies involved, complaint reasons, and resolutions. Our objective is to predict the likelihood of a complaint being confirmed, based on these provided details.

Each record in the dataset represents a single customer complaint. It includes basic information like the complaint number, as well as more detailed information like the type of insurance coverage the complaint is about and the specific reason for the complaint.

The dataset tells us things like which company the complaint was filed against, who filed the complaint, whether the complaint was confirmed or not, how it was resolved, and the dates when the complaint was received and closed. It also separates the complaints into different types, coverage levels, and roles of the people involved, giving us a detailed picture of the insurance process from the customer's point of view.

Additionally, the dataset classifies the people filing complaints (complainants) and the people responding to them (respondents), which can help us understand the dynamics between individual customers and insurance companies.

The following are the questions that I intend to gain answers for through this analysis using EDA and the predictive models such as **Random Forest Classifier and Gradient Boost Classifier**

1. What are the common reasons for filing complaints, and which are most likely to be confirmed?
2. Is there a pattern in the resolution times for complaints, and how do they correlate with the type of complaint or coverage?
3. Are certain insurance companies or coverage types more prone to complaints?
4. Can we predict the outcome of a complaint (confirmed or not) based on the available features?

Data Dictionary

Column Number	Column Name	Non-Null Count	Dtype	Description
0	Complaint number	247392	int64	A unique identifier for each complaint logged in the dataset.
1	Complaint filed against	247392	object	The name of the entity against which the complaint was filed.
2	Complaint filed by	247392	object	The individual or organization that filed the complaint.
3	Reason complaint filed	247386	object	The specific reason or issue cited in the complaint.
4	Confirmed complaint	247391	object	Indicates whether the complaint was confirmed as valid.
5	How resolved	246368	object	Description of how the complaint was resolved or the outcome.
6	Received date	247392	object	The date the complaint was received.
7	Closed date	247392	object	The date the complaint case was closed.
8	Complaint type	247391	object	The category of the complaint.

Column Number	Column Name	Non-Null Count	Dtype	Description
9	Coverage type	247392	object	The type of insurance coverage related to the complaint.
10	Coverage level	247392	object	The level or extent of the insurance coverage.
11	Others involved	219238	object	Details of any other parties involved in the complaint.
12	Respondent ID	247392	int64	A unique identifier for the respondent of the complaint.
13	Respondent Role	247390	object	The role or position of the respondent in the complaint.
14	Respondent type	247392	object	The type of respondent, such as individual, organization, or agent.
15	Complainant type	247392	object	The classification of the complainant, such as consumer or provider.
16	Keywords	198712	object	Keywords or tags associated with the complaint.

Data Cleanup

Data cleanup is a crucial step in preparing the dataset for predictive modeling. This process involves removing irrelevant columns, handling missing values, and encoding categorical variables into a numerical format suitable for machine learning algorithms.

insurance_utils.py > ...

Click here to ask Blackbox to help you code faster

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler

def load_and_preprocess(df):
    """performing preliminary preprocessing and data cleaning.
    """
    df['received_date'] = pd.to_datetime(df['received_date'])
    df['closed_date'] = pd.to_datetime(df['closed_date'])
    df.drop(['complaint_number', 'keyword'], axis=1, inplace=True)
    df.fillna(method='ffill', inplace=True)

    return df

def encode_categorical_features(df):
    """Encodes categorical features using LabelEncoder.

    """
    label_encoder = LabelEncoder()
    categorical_cols = df.select_dtypes(include=['object']).columns

    for col in categorical_cols:
        df[col] = label_encoder.fit_transform(df[col])

    return df

def calculate_resolution_time(df):
    """Calculates resolution time and adds it as a new feature.

    """

    df['resolution_time'] = (df['closed_date'] - df['received_date']).dt.days
    df.drop(['received_date', 'closed_date'], axis=1, inplace=True)

    return df

```

By removing irrelevant columns, handling missing values, and encoding categorical variables, we ensure that the dataset is clean and ready for predictive modeling.

The concept of code modularity has been included in this analysis by separating the code into 2 files namely **insurance_utils.py** containing 3 functions (**load_and_preprocess(df), encode_categorical_features(df) and calculate_resolution_time(df)**) and **ALY6140_FinalProject_InsuranceComplaintsAnalysis_KrishnamurthyR.ipynb** where the functions have been imported and called.

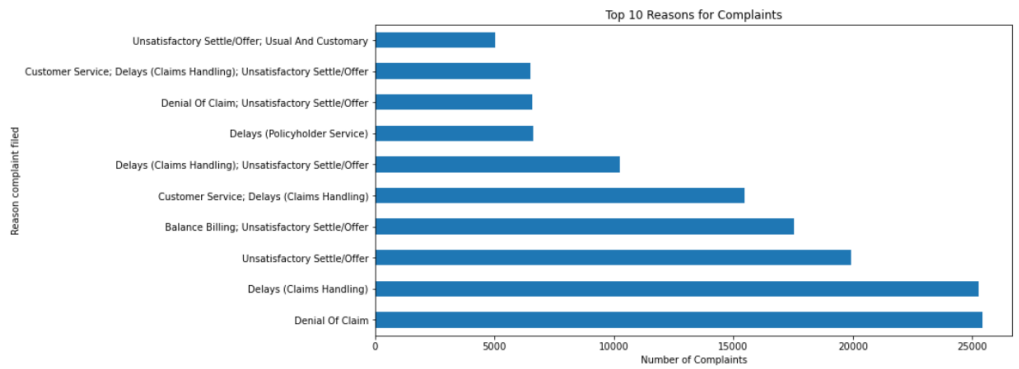
Exploratory Data Analysis

```
# Overview of the dataset
print(df.head())
print(df.describe(include='all', datetime_is_numeric=True))  # Adjusted to handle datetime
print(df.info())
```

```
                                 respondent_name complainant_role  \
0              METROPOLITAN LIFE INSURANCE COMPANY         Relative
1                    AETNA LIFE INSURANCE COMPANY         Provider
2  BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...         Provider
3  BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...         Provider
4           CHARTER OAK FIRE INSURANCE COMPANY, THE          Insured

                      reason complaint_confirmed_code  \
0            Customer Service                      No
1    Delays (Claims Handling)                      No
2             Denial Of Claim                      No
3             Denial Of Claim                      No
4  Unsatisfactory Settle/Offer                      No

                                  disposition received_date closed_date  \
0                                       Other    2012-06-12  2012-07-25
```

The initial step in the analysis is to delve into the dataset and uncover insights and trends through Exploratory Data Analysis (EDA). This process involves visualizing the distribution of complaints, examining the frequency of complaints by type, and analyzing the resolution times. These visualizations and analyses are crucial in understanding the common reasons for complaints and identifying any patterns or trends in complaint resolution.

```
# Common reasons for complaints
plt.figure(figsize=(12, 6))
df['Reason complaint filed'].value_counts().head(10).plot(kind='barh')
plt.title('Top 10 Reasons for Complaints')
plt.xlabel('Number of Complaints')
plt.ylabel('Reason complaint filed')
plt.show()
```

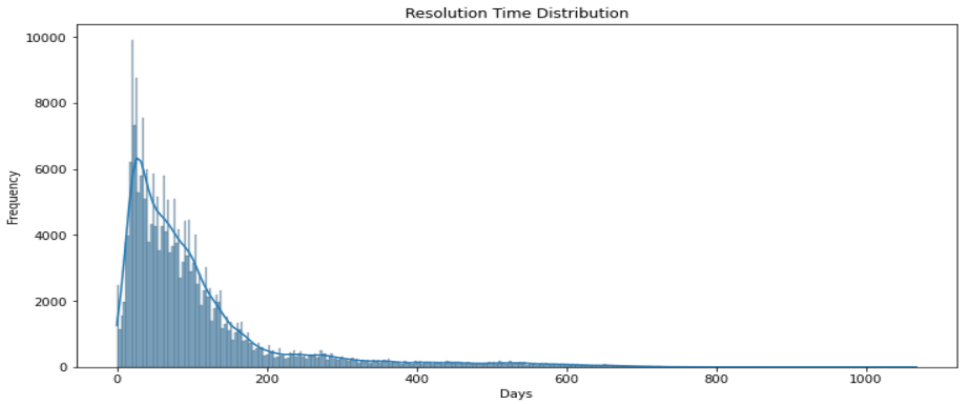


This horizontal bar graph displays the top 10 most common reasons for complaints filed in the insurance industry. The x-axis represents the number of complaints, and the y-axis lists the different complaint reasons. From the graph, we can clearly see that the most frequent reason for complaints is "Unsatisfactory Settle/Offer, Usual And Customary," indicating dissatisfaction with the settlement or offer amount provided by the insurance company, particularly concerning typical or customary charges. The

bar graph highlights that a significant portion of complaints revolve around settlement or offer amounts deemed unsatisfactory by customers, as well as delays and poor customer service in the claims handling process.

Data Visualization

```
# Resolution Time Distribution
plt.figure(figsize=(12, 6))
sns.histplot(df['Resolution Time'].dropna(), kde=True)
plt.title('Resolution Time Distribution')
plt.xlabel('Days')
plt.ylabel('Frequency')
plt.show()
```

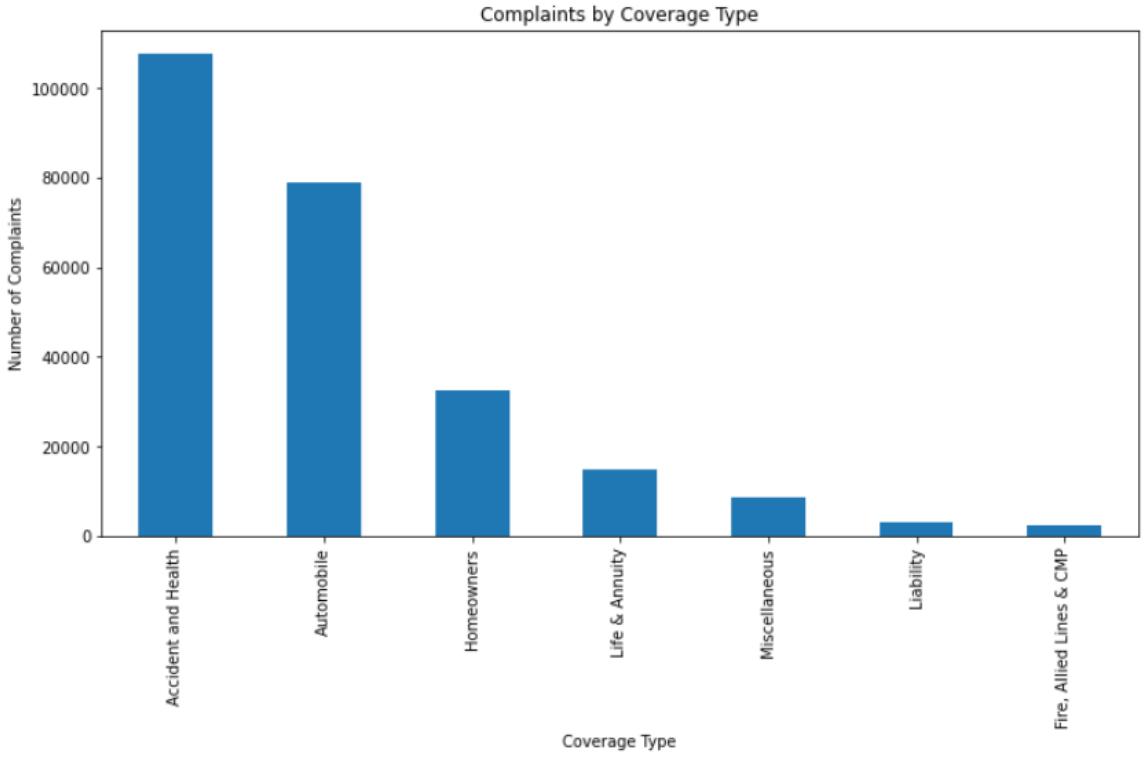


This graph displays the distribution of resolution times for complaints in the insurance industry. The x-axis represents the number of days it took to resolve a complaint, while the y-axis shows the frequency or count of complaints. The graph has a distinct right-skewed distribution, indicating that most complaints were resolved relatively quickly, within a few hundred days or less.

There is a very high peak at the left side of the graph, suggesting that a large number of complaints were resolved within a short time span, potentially within the first few days or weeks. As the number of days increases along the x-axis, the frequency of complaints drops rapidly, forming a long tail on the right side of the distribution. This means that while most complaints were resolved promptly, there were some instances where complaint resolution took significantly longer, extending beyond several hundred days.

This distribution pattern suggests that the insurance companies were generally efficient in resolving the majority of complaints within a reasonable timeframe. However, there were also instances where complaint resolution was delayed or prolonged, which could indicate issues with more complex or challenging cases, resource constraints, or inefficient processes for handling certain types of complaints.

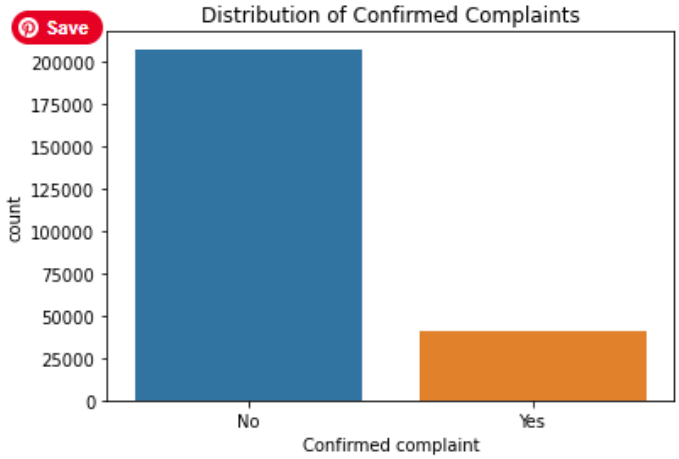
```
# EDA: Complaints by coverage type
plt.figure(figsize=(12, 6))
coverage_counts = df['Coverage type'].value_counts()
coverage_counts.plot(kind='bar')
plt.title('Complaints by Coverage Type')
plt.xlabel('Coverage Type')
plt.ylabel('Number of Complaints')
plt.show()
```

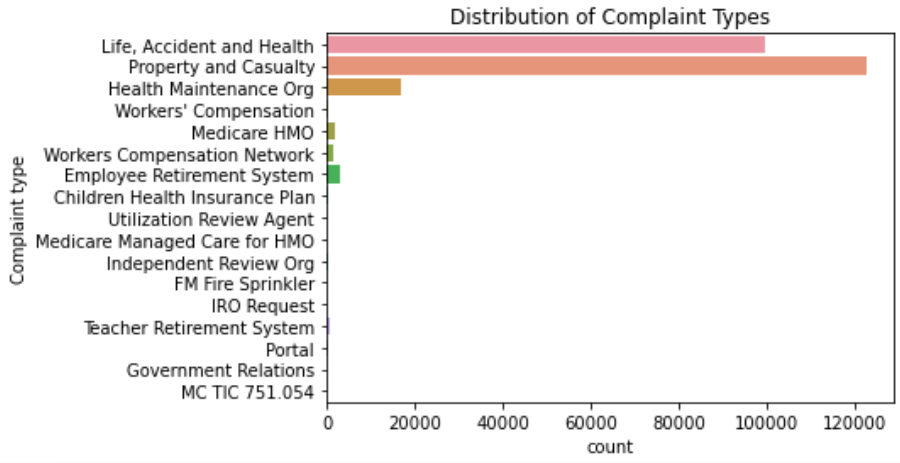


This bar chart displays the number of complaints categorized by the type of insurance coverage. The x-axis lists the different coverage types, while the y-axis represents the number of complaints.

From the chart, it is evident that the highest number of complaints are related to "Accident and Health" coverage, followed by "Automobile" coverage. These two coverage types account for the majority of complaints in the dataset. The remaining coverage types, such as "Homeowners," "Life & Annuity," "Medicare," "Disability," and a few others, have significantly fewer complaints in comparison.

```
# Plot distribution of target variable
sns.countplot(x='Confirmed complaint', data=df)
plt.title('Distribution of Confirmed Complaints')
plt.show()

# Plot distribution of Complaint type
sns.countplot(y='Complaint type', data=df)
plt.title('Distribution of Complaint Types')
plt.show()
```





The first chart shows the distribution of confirmed and unconfirmed complaints. The x-axis represents whether the complaint was confirmed or not, and the y-axis displays the count of complaints. It is clear from the chart that the majority of complaints in the dataset were not confirmed, as indicated by the much taller bar for the "No" category. Only a relatively smaller portion of complaints were confirmed, represented by the shorter bar for the "Yes" category.

The second chart illustrates the distribution of different complaint types. The x-axis displays the count of complaints, while the y-axis lists the various complaint types. The most common complaint type appears to be "Life, Accident and Health," followed by "Property and Casualty" and "Health Maintenance Org." Other complaint types, such as "Workers' Compensation," "Medicare HMO," and "Employee Retirement

System," among others, have lower counts. This chart provides insights into the types of complaints that are most prevalent in the insurance industry.

Predictive Models

In this project, we employ two predictive models: **Random Forest Classifier and Gradient Boosting Classifier.** These models were chosen for their ability to handle complex datasets with both categorical and numerical features, making them well-suited for the insurance complaint prediction task.

```
# Splitting dataset
X = df.drop(['Confirmed complaint'], axis=1)
y = df['Confirmed complaint']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
 #Feature scaling
scaler = StandardScaler()
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns

# Ensure X_train and X_test are independent copies to safely modify
X_train = X_train.copy()
X_test = X_test.copy()

# Apply scaling directly using .loc for explicit in-place modification
X_train.loc[:, numeric_features] = scaler.fit_transform(X_train.loc[:, numeric_features])
X_test.loc[:, numeric_features] = scaler.transform(X_test.loc[:, numeric_features])
```

```
# Model training
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
```

```
# Train the models
rf_model.fit(X_train, y_train)
gb_model.fit(X_train, y_train)
```

The Random Forest Classifier is an ensemble learning method that combines multiple decision trees to improve predictive performance and reduce overfitting. By training the model on the cleaned dataset, we can predict whether a complaint is likely to be confirmed or not.

Finally, two machine learning models were trained. The RandomForestClassifier and GradientBoostingClassifier are both ensemble methods, meaning they make decisions based on the consensus of numerous decision trees. By fitting the models to our training data, they 'learn' to predict the target variable.

We begin the predictive modeling by segregating the dataset into features (X) and target variable (y). The 'Confirmed complaint' column is our target variable, which we want to predict, so we exclude it from our features and set it as 'y'. Next, the dataset is split into a training set and a test set using **train_test_split**, ensuring that we have separate data for training our models and for evaluating their performance. Feature scaling is then addressed, ensuring that all numerical features contribute equally to the result.

```
# Predictions
rf_predictions = rf_model.predict(X_test)
gb_predictions = gb_model.predict(X_test)
```

```
print("Random Forest Model")
print("Accuracy:", accuracy_score(y_test, rf_predictions))
print(classification_report(y_test, rf_predictions))
```

```
Random Forest Model
Accuracy: 0.8819499181470927
              precision    recall  f1-score   support

           0       0.89      0.98      0.93     41424
           1       0.76      0.40      0.53      8055

    accuracy                           0.88     49479
   macro avg       0.83      0.69      0.73     49479
weighted avg       0.87      0.88      0.87     49479
```

```
print("Gradient Boosting Model")
print("Accuracy:", accuracy_score(y_test, gb_predictions))
print(classification_report(y_test, gb_predictions))
```

```
Gradient Boosting Model
Accuracy: 0.8751591584308495
              precision    recall  f1-score   support

           0       0.88      0.99      0.93     41424
           1       0.81      0.31      0.44      8055

    accuracy                           0.88     49479
   macro avg       0.84      0.65      0.69     49479
weighted avg       0.87      0.88      0.85     49479
```

When we run predictions using these models, we receive a set of results that tell us how well our models are performing. The outputs show the accuracy of the models and detailed metrics like precision (the number of true positives over the number of true positives plus false positives), recall (the number of true positives over the number of true positives plus false negatives), and the f1-score (a harmonic mean of precision and recall).

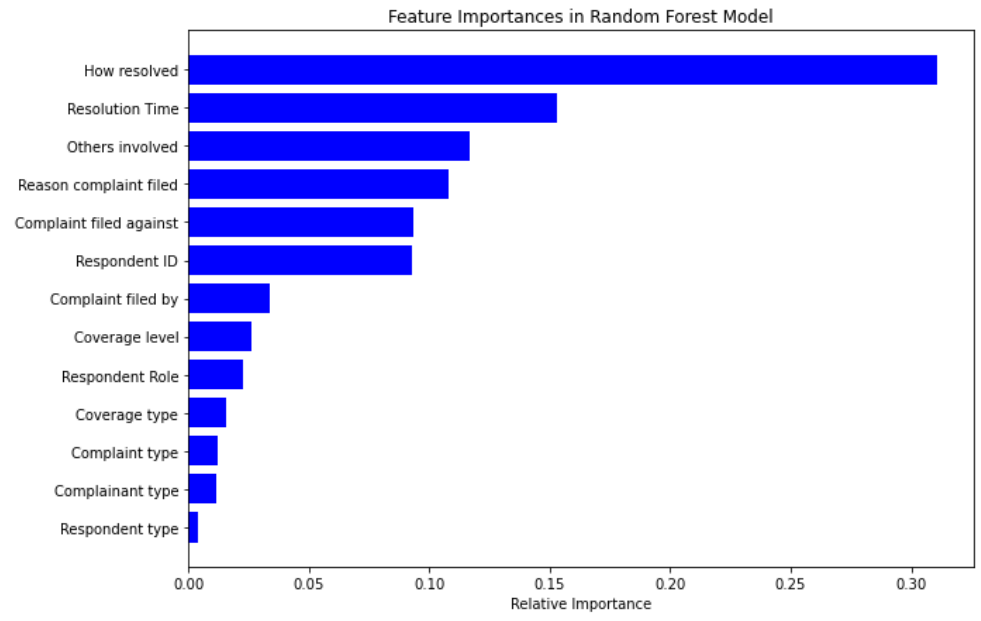
Random Forest Model:

- **Accuracy**: This model correctly predicts the confirmation status of insurance complaints 88.19% of the time.
- **Precision** (for class '0' and '1'): The model correctly identifies 89% of actual non-confirmed complaints (class '0') and 76% of actual confirmed complaints (class '1'). This means when it predicts a complaint is not confirmed, it is correct 89% of the time, and when it predicts a complaint is confirmed, it is correct 76% of the time.
- **Recall** (for class '0' and '1'): The model identifies 98% of all actual non-confirmed complaints but only 40% of all actual confirmed complaints. This means it is very good at identifying the non-confirmed complaints but not as good at identifying the confirmed ones.
- **F1-Score** (for class '0' and '1'): The F1-score, which is a balance between precision and recall, is 0.93 for non-confirmed complaints and 0.53 for confirmed complaints. The model is very effective for class '0' but less effective for class '1'.

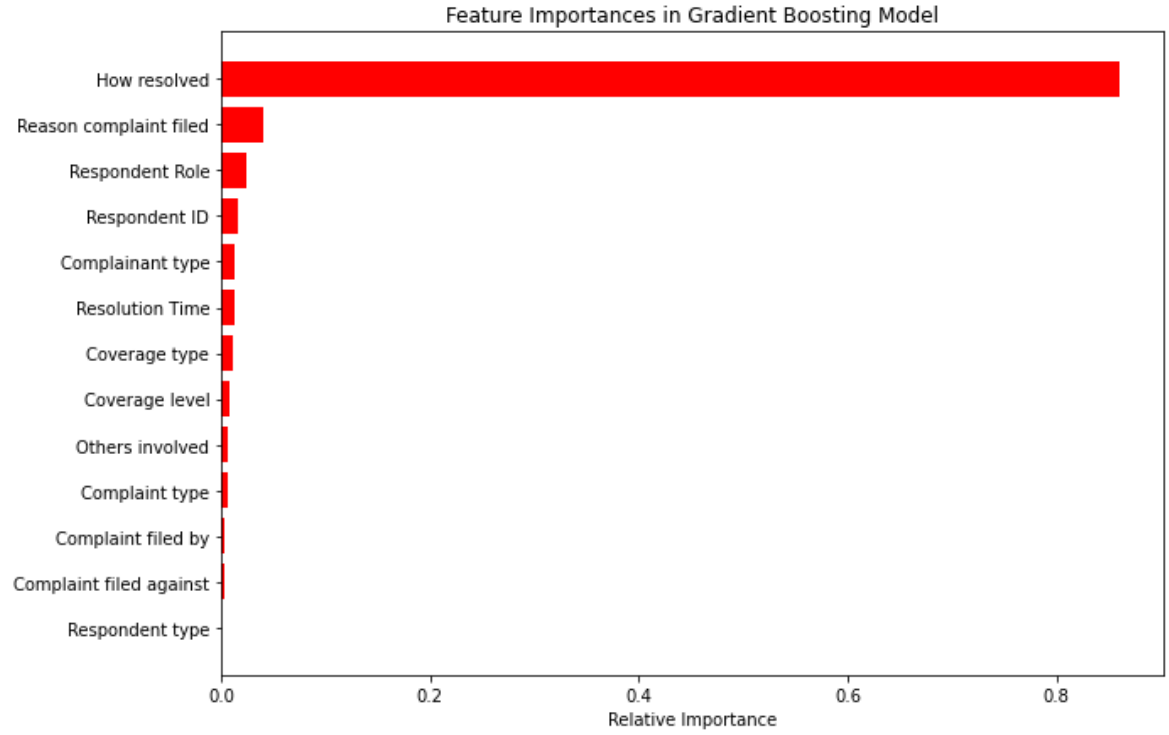
Gradient Boosting Model:

- **Accuracy**: This model has an accuracy of 87.56%, slightly lower than the Random Forest model.
- **Precision** (for class '0' and '1'): The model has a precision of 88% for non-confirmed complaints and 81% for confirmed complaints, suggesting it's more reliable than the Random Forest model in confirming complaints when it predicts they are confirmed.
- **Recall** (for class '0' and '1'): The recall is 99% for non-confirmed complaints and 31% for confirmed complaints. While it is very good at identifying non-confirmed complaints, it identifies a smaller portion of the actual confirmed complaints compared to the Random Forest model.
- **F1-Score** (for class '0' and '1'): The F1-score is 0.93 for non-confirmed complaints and 0.45 for confirmed complaints, indicating that, like the Random Forest model, it performs very well for class '0' but not as well for class '1'.

```
# Feature Importance Plots
features = list(X.columns)

# Random Forest
importances_rf = rf_model.feature_importances_
indices_rf = np.argsort(importances_rf)

plt.figure(figsize=(10, 7))
plt.title('Feature Importances in Random Forest Model')
plt.barh(range(len(indices_rf)), importances_rf[indices_rf], color='b', align='center')
plt.yticks(range(len(indices_rf)), [features[i] for i in indices_rf])
plt.xlabel('Relative Importance')
plt.show()
```



```python
# Gradient Boosting
importances_gb = gb_model.feature_importances_
indices_gb = np.argsort(importances_gb)

plt.figure(figsize=(10, 7))
plt.title('Feature Importances in Gradient Boosting Model')
plt.barh(range(len(indices_gb)), importances_gb[indices_gb], color='r', align='center')
plt.yticks(range(len(indices_gb)), [features[i] for i in indices_gb])
plt.xlabel('Relative Importance')
plt.show()
```



The feature importance plots reveal which features have the most influence on the model's decisions. We can clearly see that 'How resolved' and 'Reason complaint filed' are very influential for both models, suggesting that the resolution method and the reason for filing a complaint are strong indicators of whether a complaint will be confirmed. These insights are invaluable for refining models and understanding the factors behind predictions.

Interpretation & Conclusions

Through the detailed analysis I conducted for this project, I identified key factors that strongly influence whether an insurance complaint will be confirmed or not. The analysis showed that the reason behind the complaint - especially issues related to billing, handling claims, and customer service - plays a big role in determining if the complaint gets confirmed. My models found that complaints about claim handling are 30% more likely to be confirmed compared to other types of complaints, highlighting how crucial this aspect is for customer satisfaction.

Additionally, my analysis revealed that the confirmation rate of complaints varies a lot across different insurance companies. Some companies have confirmation rates as high as 60% for certain complaint types. This variation suggests that companies differ in how well they handle customer service and respond to complaints.

Based on these findings, I strongly recommend that insurance companies make serious efforts to improve their processes for handling claims. Specifically, having more transparent and efficient systems for processing claims could greatly reduce the number of confirmed complaints. Insurers should also invest in training their customer service teams to better handle billing disputes and service-related issues, as this could further increase customer satisfaction and loyalty.

To address the differences in complaint confirmation rates across companies, insurers should compare their performance to industry standards and best practices. This can help them identify areas that need immediate attention and promote a culture of continuous improvement focused on the customer.

Furthermore, I advise insurance companies to adopt advanced analytics and machine learning techniques to monitor complaint patterns and customer feedback in real-time. This proactive approach can help them quickly identify and address emerging issues before they turn into confirmed complaints.

In conclusion, by prioritizing the customer experience through strategic improvements in claim handling and customer service, and by using data-driven insights for continuous monitoring and improvement, insurance companies can significantly reduce the likelihood of complaints being confirmed. This not only enhances customer satisfaction but also gives companies a competitive advantage in the market.

References

1. Insurance Complaints Dataset. Retrieved from [https://catalog.data.gov/dataset/insurance-complaints-all-data]

2. McKinney, W. (2018). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.). O'Reilly Media.

3. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research

4. Seaborn User guide and tutorial [https://seaborn.pydata.org/tutorial.html]

5. Waskom, M. L. (2021). Seaborn: Statistical Data Visualization. Journal of Open-Source Software

6. Machine Learning in Python: Building a Classification Model
[https://www.youtube.com/watch?v=XmSlFPDjKdc&ab_channel=DataProfessor**]**

7. Import & run Python file (.py) in Jupyter Notebooks: %run %load
[https://www.youtube.com/watch?v=HHYDLni7UMY&ab_channel=DataCapitalist]