**Problem Description:**

Problem consists of predicting the type of the document based on hashed document data. This problem is multi class classification problem with 14 target variables and hashed document words.

**List of major concerns/assumptions:**

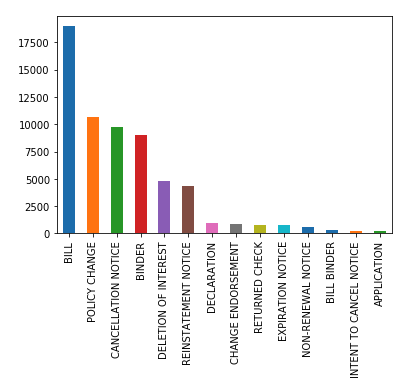
The dataset words are the output of OCR stage of data pipeline. Hence words contain stop words which are also hashed and words in the document are in order

**Data Description:**

The data consists of 14 document labels and hashed document content.

**Exploratory Data Analysis:**

Below is the count of documents for each document label. I observed BILL and POLICY Change contains a greater number of documents whereas INTENT TO CANCEL NOTICE and APPLICATION has a smaller number of documents



As the document content is hashed hence I could not perform more operations on real data like removing stop words, stemming, and lemmatization.

**Missing Values in Data:**

There are 45 records where the document content is missing

$df.isna().sum()

label 0

doc 45

**Analysis Plan:**

**Feature Engineering:**

I used TFIDF (Term Frequency Inverse Document Frequency) to create feature vectors where is used Sklearn TFIDF vectorizer and transformer to fit all training, test and validation vectors in same dimensions.

Encoded the document labels to make categorical values discrete using LabelEncoder

**Explanation of Modeling Choice:**

The first natural step in creating basic logistic model will help you to understand all of 55 predictor variables and this model was used as a stepping stone to other models. In random forest model we found the most significant variable in based of mean decrease in accuracy as well as contributing to reduction in node impurity. We selected Gradient boosting technique as it helps us in finding variable importance plot which will help us to see the best predictor variables.

I split the training data into three parts with the ration of 65, 15 and 15, first part is to train the model, second part of the data is to validate the model, and third part of the data to test the model. If there is some overfitting or underfitting validation and testing will perform abnormal

**Modeling Technique and its Strength, Weakness:**

Logistic regression helps to understand the coefficients and significance of the variables. The exponential of coefficients corresponds to odd ratios for the given factor. Logistic regression requires that all variables are independent of each other if there is any correlation between variables and then the model will tend to overweight the significance of those variables

Random forest runs fast, and it is good at dealing with unbalanced data. To do classification with Random Forest, it’s cannot predict beyond the range in the training data, and that there may be chances of over-fit datasets that are particularly noisy. But the best test of any algorithm is how well it works upon your own data set. We also chose this model because it helps us in finding the feature importance plot which is bases for the understanding the features and coming up with performing feature engineering.

I tried other various models like Support Vector Machine (SVM), Decision Trees, and K-nearest neighbors so that I can compare model performance metrics for the better accuracy. Hyperparameter tuning definitely help to make model better for random forest and SVM.

**Metric Evaluation for Machine Learning Models:**

|  |  |  |
| --- | --- | --- |
| Method | Validation Data Accuracy | Test Data Accuracy |
| Logistic Regression | 0.86 | 0.862 |
| Decision Tree | 0.77 | 0.77 |
| Random Forest | 0.8218 |  |
| K Nearest Neighbors | 0.8004 |  |
| Random Forest |  |  |

**Web Page Development:**

* I used Flask which is a micro frame work written in python.
* Restful API for the web service and it returns json formatted output
* Used HTML/CSS/JavaScript/Bootstrap to develop basic webpage where it contains text box and submit button to take an action

**Deployment:**

* Deployed web application in Amazon Webservice
* I generated a pickle file for trained logistic regression classifier(machine\_learning\_lr.pkl) which helped me to deploy in production. I could not upload pickle file in git as it is having more than 100 MB
* Below is the link for accessing web application hosted in AWS

<http://18.191.6.105/>

**Execution Instructions:**

1. **Type 1 Simple Program execution:**

**Step1**: Download the pickle file from below location and store in classification folder. Please let me know if you don’t have access.

https://drive.google.com/file/d/1Q4u2sk9KXsmTkVhOtF8I-iBSQs8tzxgz/view?usp=sharing

**Step2:** Install requirements.txt file

Pip3 install -r requirements.txt

**Step3:** Run below command to execute the python file

Python3 classification.py

1. **Type 2:** To access web application, you may open below link to test the document type. This web link contains text box and submit buttion

<http://18.191.6.105/>

1. As I used REST API, you may execute below command predict document type and confidence

$curl http://18.191.6.105/ -d "Document Content= words”

Note: To run against your webservice you may change the above web service URL

**Challenges Faced:**

1. As the document content is hashed I could not able to do the feature engineering with best features to train a model
2. In order to deploy in AWS, I tried various methods Zappa, Nginix server but I had experienced some difficulties on resolving few errors.

**Improvements:**

1. Hyper Parameter tuning for support vector machine and XG Boost model definitely improve the accuracy, but it needs GPU’s to run the program
2. Removing stop words, Stemming and Lemmatization will also improve the accuracy if we know actual data.
3. TFIDF vector created big dimension vector where most of the columns are zero if we delete the most common columns, again we can make model smarter. But handling huge dimension vectors again we need GPU’s.