**DETAILED PROJECT REPORT**

**INSURANCE FRAUD DETECTION**

**ABOUT PROJECT**

The name of the project is Insurance Fraud Detection system.

Problem Statement:

Build such an application which can correctly identifies which insurance claims are fraudulent and which insurance claims are genuine. Whatever claims are identified by the system are fraudulent claims, they won’t be approved and in one click it will go through manual inspection.

This problem statement is a Supervised Machine Learning approach.

**Features:**

* Able to work with large datasets
* Works even with PC’s, you don’t need high-end systems

**DURATION OF THE PROJECT**

To complete this project efficiently, our team took 6months of time in brainstorming for the right approach, documentation of the project and building the project. For testing and checking for bugs and other faults we took 18 days of time. So, in total the project took roughly about 7months of time to complete.

**TEAM SIZE AND DISTRUBUTION**

The team consist of:

* 1 Product Manager
* 1 Solution Architect
* 1 Lead
* 2 Dev-Ops engineer
* 2 QA engineers
* 1 UI developers and
* 2 Python Developers
* 3 Data Scientists

**DATA DESCRIPTION**

Data consist of 40 columns out of which Fraud Report column is the target column where Y denotes Fraud Reported and N denoted Fraud Not-Reported. Other columns contain numerical and categorical data as well. The data set contains missing values which are represented with a question mark (?).

**LOG MAINTAINING**

Logs are important to maintain because they make the debugging part easy when it comes to bug fixing part. Keeping this in mind, we tried to log every possible step that are performed in the training and predicting stages of the project.

We have built a simple class for maintain the logs, the class performs writing the date, time and log message into a file.

The process follows:

**DATA VALIDATION, TRANSFORMATION AND PRE-PROCESSING**

Data validation includes several steps such as:

* File format and file name convention check is done by creating a manual regex creation.
* Length of the time stamp and date stamp is done.
* Number of rows and columns is checked.
* Number of files sent is checked.

Data Transformation involved various techniques and steps. The following steps are performed in data transformation:

* The initial check for the data type is done.
* Missing values are replaced by null vales.
* Double quotes to the string type of data is also checked.

Data pre-processing involved various techniques and steps. The following steps are performed in data preprocessing:

* If null-values are equal to the column length, we move that file to bad files folder and if not, we imputed the null-values using KNN imputation techniques.
* If the data is not-normal, is converted into normal distribution.

A fully-fledged data transformation and loading pipeline is used to automate the data preprocessing.

**FAILURE CASES**

For all the files that are failed to preprocess are moved into bad files folder and finally into archive folder. Further the files were sent to the client for discussion.

**TECHINEQUES FOR TRAINING THE MODEL**

After receiving all the good files from the preprocessing stage, the columns with zero standard deviations is dropped as they will not contribute anything towards the prediction. After dropping the relevant columns, the feature and the label columns are separated for the training purpose. After separation, we check for null-value presence in the feature columns, if yes, we impute the null values using kmean imputer. This is done to increase the model accuracy and to avoid errors that can be triggered while clustering or training. After imputation, we again check for columns with zero standard deviation in the feature columns, and if present, we drop the respective column.

After these above processes are done successfully, we go for the clustering approach for the feature columns. Clustering is done to further increase the model accuracy and to better train the model. Here, in our approach, we have used Kmeans Clustering technique for creating clusters. The number of clusters to be formed is found using the elbow-plot method. After dividing the features through clustering into the obtained number of clusters, we then pass each cluster for training to find the best model. The number of models will be equal to the number of clusters formed.

Each cluster data will further be divided into cluster features and cluster labels. These will again be divided into train and test data sets for feature and label data. These will be used to find the best model for the data set and will be saved as .sav or pickle file format to be used in the prediction stage.

**DEVELOPMENT AND PRODUCTION**

The necessary development of the entire project was done in one development server. The project was running fine with the training phase and although a higher system performance will enhance the speed of building the model faster. We used the development server to create the model and save the model. The same saved model was deployed to the cloud UAT and production environments.

**HYPERCARE**

The postproduction maintenance was done for the possible bugs that were encounter by the client by the QA team. The process includes mostly with the file format, naming convection and the data type in the files.

**ALL TECH**

The project includes a lot of technologies such as:

* SQLite for the database management
* SkLearn for the machine learning approaches
* Flask APIs for hosting and dashboard monitoring purpose
* Heroku platform for hosting the app