***ML Insurance Fraud Detection***

1. **Problem Statement**

* To build classification model to predict whether the insurance claim made by the customer are fraud or not.
* Companies can use this type of classification to identify whether the customer trying to claim the insurance amount is genuine one or just trying to fake the incident for the insurance money.

1. **Description of data**

* Dataset contains continuous values and categorical values.
* Columns inside the dataset are:

1. Months\_as\_customer: number of months since the customer opened insurance account
2. Age: age of the customer
3. policy\_number: policy number of the customer
4. policy\_bind\_date: policy start date
5. policy\_state: state name where policy is brought
6. policy\_csl: combined single limits – amount of bodily injury will be covered from the total damage (eg: 500/1000)
7. policy\_deductable: amount paid out of pocket by the policy holder before insurance provider will pay any expenses.
8. Policy\_annual\_premium: yearly premium for the policy
9. Umbrella\_limit: umbrella insurance policy is extra liability insurance coverage that goes beyond the limits of the insureds homeowners, auto or watercraft insurance. It provides an additional layer of security to those who are at risk of being sued for damages to other peoples property or injuries caused to others in an accident.
10. Insured\_zip: zip code where policy is registered
11. Insured\_sex: denotes the persons gender
12. Insured\_education\_level: the highest education level of the policy holder
13. Insured\_occupation: occupation of the policy holder.
14. Insured\_hobbies: hobbies of policy holder.
15. Insured\_relationship: dependents on the policy holder
16. Capital\_gain: monitory gains by the person
17. Capital\_loss: monitory loss by the person
18. Incident\_date: the date when the incident happened
19. Incident\_type: type of incident
20. Collision\_type: type of collision that took place
21. Incident\_severity: severity of the incident
22. Authorities\_contacted: which authority was contacted
23. Incident\_state: the state in which incident occurred
24. Incident\_city: city in which incident took place
25. Incident\_location: street in which the incident took place
26. Incident\_hour\_of\_day: hour at which the incident took place
27. Number\_of\_vehicle\_involved: cars involved in the accident
28. Property\_damage: if any property damage was done.
29. Bodily\_injuries: how many bodily injury persons suffered
30. Witnesses: number of witnesses present
31. Police\_report\_available: whether a police report is available or not
32. Total\_claim\_amount: amount claimed by the person from insurance company.
33. Injury\_claim: amount claimed for the bodily injuries
34. Property\_claim: amount claimed for the property damages
35. Vehicle\_claim: amount claimed for the vehicle damages
36. Auto\_make: vehicle brand and type name
37. Auto\_model: model number of the car
38. Auto\_year: year the model of car purchased
39. fraud\_reported: Target label, whether fraud is reported or not.
40. **Application architecture and module division**

* Bigger problem: How to classify the data to decide whether the claim is true or false?
* Better to break down the development into small parts so that changes to be made in end of module doesn’t affect the other modules. Multiple members part of the project so better to divide the modules
* Broker into small subgroups:

1. How to read the data
2. How to validate the data
3. How to do data preprocessing and how to train a model on the data
4. How to do hyperparameter tuning for the model

* **Step 1**: Data ingestion

1. Data for training - client provides or stores the data needed at a particular location, aggregate multiple data sources
2. Data validation – discuss with client the datatype of variables, number of variables, whether any columns contain only null values
3. Data transformation – missing values conversion to null, categorical values in commas or “” and maybe not accepted in DB
4. Data insertion in DB – after transformation insert the data inside the database for further development

* **Step 2**: Training Pipeline / Step

1. Export the data in CSV from DB and csv acts as train data
2. Data preprocessing – perform EDA, check if there are any null values present, convert categorical values into numerical values, if data is imbalanced or normalized
3. Data clustering - to increase the accuracy of the model we divide the data into individual clusters and build model for each cluster separately
4. Hyperparameter tuning - to increase the performance of the individual model selected for each cluster
5. Model saving – save the model for each cluster individually

* **Step 3:** Deploy on cloud

1. Create metadata for pushing the app onto the cloud server
2. Start and test the application

* **Step 4**: Prediction Pipeline

1. Data validation – discuss with client the datatype of variables, number of variables, whether any columns contain only null values
2. Data transformation – missing values conversion to null, categorical values in commas or “” and maybe not accepted in DB
3. Data insertion in DB – after transformation insert the data inside the database for further development
4. Export the data in CSV from DB and csv acts as train data
5. Data preprocessing – perform EDA, check if there are any null values present, convert categorical values into numerical values, if data is imbalanced or normalized
6. Data clustering - to increase the accuracy of the model we divide the data into individual clusters and build model for each cluster separately
7. Call the model for specific cluster number stored
8. Make prediction and export the prediction in a csv file

* **Step 5:** Model retraining

1. When new patterns detected these changes must be aggregated to the model.
2. Provide the prediction + train data to the model for retraining
3. Logging and monitoring framework
4. **Code**:

Main .py -🡪 1. Validation step – read data, validation, transformation, insert into DB, export

to csv file.

* 2. Training – read train data, data preprocessing, data clustering, model finding,

Model tuning, deployment

* 3. Prediction – validation,

Prediction – model saved loaded into memory and make predictions

* Data for training: User provides different training batch files
* Synchronize logging, asynchronous – code doesn’t wait for completing the logging faster and individual
* Data Validation: Whether the data sent by the client is valid or not as per requirement given. File name is correct or not based on agreement, if we reject the data push into bad folder else put in good data folder. We use the schema files created. We delete the good and bad directory as the good data is stored inside the database. Check the length of the timestamp given in the file name format, check the number of columns present in the data, check if any column contains just all column values as null values.
* Data Transformation: DB doesn’t accept Nan value so we transform to NULL using fillna. Add quotes to all values present in the categorical values for insertion into the DB.
* Data Insertion in DB: Put the good raw data inside the DB using the datatypes given inside the schema. Export the good raw data table into csv.
* Data preprocessing: Use the exported csv file as input. Perform EDA to understand the data and what processing is needed for the data.

Strip or remove the white spaces in categorical values.

Replace the “?” with NaN in the dataset and check number of missing values.

Drop columns not necessary for further classification method. Eg. Policy number, policy state, zipcode don’t contribute to know whether its fraud or no (Based on the domain knowledge we drop variables not important).

Impute categorical missing values – using sklearn categoricalImputer (imputing with the mode value)

Separate categorical and numerical variables. Decide whether to use getdummies or custom mapping to convert categorical variable present. We perform custom mapping eg.’100/500’: 1, ‘College’:1 Map according to order for the education level, injury level

Combine numerical and categorical data after analysis and separate the X and target y

Perform visualization for checking the distribution of target based on target y. Check scatterplot, distplot, heatmap for correlation

Store null values information inside a file for handling it in future

* Model Training: Fetch the data and perform the preprocessing decided after doing the EDA. Separate the target and features. Perform clustering on the cleaned data and create clusters. We can apply cluster specific algorithms for better performance. Loop individual clusters and apply different models. Perform standard scaler on the input data.

Logistic regression, DT, RF, SVM, NB, KNN, XGBoost algorithms can be used for classification.

* We perform hyperparameter tuning for the selected algorithms and check the performance of the model. Then save the model for further prediction. We used SVM and XGBoost. If same label classes present then we use the accuracy score else we can use the roc\_auc\_score for the check. Save the model for each cluster created.
* Prediction –

1. perform validation – for filename, no of columns, all of null values
2. perform transformation – replace nan, insert into db, export csv as input
3. data preprocessing – missing values and imputation, std == 0 dropped, drop unnecessary columns
4. Perform clustering to determine which cluster it belongs to using Kmeans.predict
5. Then based on cluster assigned use the respective model for each individual cluster. Reassign the encode class category back on the predicted file.

* Index.html – default for every browser it call webage it returns httpget

Render template to display respective html pages.

* Deployment for cloud 🡪

1. Requirements.txt – import of packages are included here, as cloud needs instruction for cloud deployment (pip freeze requirements.txt)
2. Using Heroku steps