Insurance Fraud Detection …….

Insurance fraud is a huge problem in the industry. It's difficult to identify fraud claims. Machine Learning is in a unique position to help the Auto Insurance industry with this problem.

In this project, the dataset has the details of the insurance policy along with the customer details. It also has the details of the accident on the basis of which the claims have been made.

Data Cleaning:

The data has to be cleaned and validated before building a model in order to make sure the model works pretty good and has minimal errors in it.

First import the data and copy it to a variable. Let it be named as df.

df.info() – Shows the count of non-null values in all the columns. We can notice if there are any missing values in any of the variables.

df.isnull().sum() – Shows the count of null values in each column

- ➔ The rows with null values can be removed, but the removal of data should be the last resort and the data loss should not exceed 8% as the best practice set to myself.
- ➔ To fill the null values, we can use the basic methods such as forward fill or backward fill. (df.fillna(method = 'ffill') , df.fillna(method='bfill'))

In this problem, there are no null values in the data set. So, we will just proceed with feature engineering and EDA

The columns with object type and needs to be converted to numerical to be used in modelling. The Date column first is converted to datetime format and day of week, months and year data is extracted from it. There are two columns with date in this data set

```
df['Date'] = pd.to_datetime(df['Date'],dayfirst = True)
```

The date type data is converted, extracted and added to the data. Few columns are continuous and Few are categorical.

EDA:

The variables in this data are continuous as well as categorical.So, we can make use of countplot, boxplot, distplot and so on..

The count of unique values in the target variable:

```
df.fraud_reported.value_counts()
```

```
N    753
Y    247
Name: fraud_reported, dtype: int64
```

It is almost in the ratio of 75 : 25 and it can be considered as a balanced target column.

To check the count of values in a specific categorical variable with target variable, we can use count plot along with hue to compare the data.

```
sns.countplot(df[# column name], hue = df[# target variable])
```

Now to check the distribution, we can use the below code:

```
sns.distplot(df[#column name for which distribution has to be checked] )
```

Since, there are many columns with continuous type of data I won't be adding the graphs here. Please visit the below github link for the ipynb file:

https://github.com/vangetiakhil/Evaluation-Projects/blob/main/Insurance.ipynb

Since, the data is not normally distributed, there is a skewness which can be due to the outliers.

To check the skewness,

```
df.skew()
```

The columns with skewness between -0.5 to 0.5 are acceptable. This does not mean the data is normally distributed. We even need to see for that.

For checking outliers,

```
sns.boxplot(df[# column name])
```

https://github.com/vangetiakhil/Evaluation-Projects/blob/main/Insurance.ipynb

Even the outliers are less.

Since, this is a fraud detection problem, every data point is considered to be important. Hence, no data is removed from this dataset.

The categorical variables need to be encoded. This can be done by the use of LabelEncoder and OneHotEncoder. The columns has unqiue values more than 2.

Instead of using LabelEncoder, I went with the use of dummies.

```
final_df = pd.get_dummies(df, columns = [all categorical columns])
```

Preprocessing:

To build a good model, all columns must be under same limits. Standardization is the option for that. We only need to standardize the independent variables. Therefore, splitting the data into target (y) and independent (x) data frames.

Now standardizing the X.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaledX = pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
```

The problem here is to predict the target variable 'fraud reported'. Since, there are two unique values in the column and they are categorical, this comes under classification problem.

The models we can use are logistic regression as there are only two output values in the target variable and decision tree classifier, SVC, K nearest neighbor classifier and so on…

The metrics we can use this problem are accuracy score, confusion matrix and classification report

After building the models and finding their accuracy score, check that with the cross val score (f1). Prefer the model with less difference.

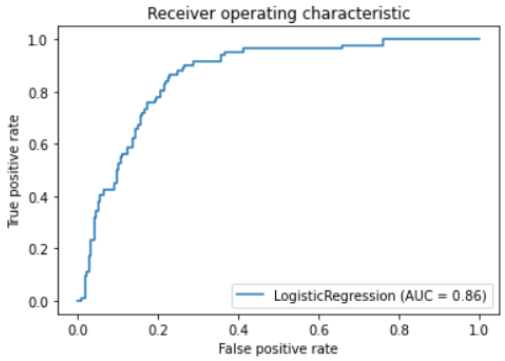
Tune the parameters using gridsearchcv for that modelling technique and build the final model:

```
gcv = GridSearchCV(mod(),param_grid = parameters, cv=5)
gcv.best_params_
```

Check the AUC by plotting the ROC curve as below:

```
plot_roc_curve(mod,xtest,ytest)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('Receiver operating characteristic')
```

```
Text(0.5, 1.0, 'Receiver operating characteristic')
```



Note :- mod in the above code represents the model finalized.

Auc –Roc curve can only be plotted for classification problems

The nearer the value of AUC to ‘1’, the better the performance of the model.

For the detailed process, please do visit the link:

https://github.com/vangetiakhil/Evaluation-Projects/blob/main/Insurance.ipynb