1) Model performance metrics (AI & ML) (1)

Classification & Regression model

↓

Risk classification

↘

predict. → 1) Low Risk
2) Medium Risk
3) High Risk

Multiclass classification

Based on → 1) Customer Age,
2) Annual Income,
3) vehicle/property age,
4) claim history
5) premium amount
6) claim amount
7) Fraudulent claim Happen (or) Not
1 (or) 0
8) policy type
9) Gender. → Not important

model Training:

to Try Different machine Learning Algorithm

get result from overfit

train Accuracy higher the

test accuracy

1) Then use SMOTE check model

is balanced or Not

Model is Imbalanced

So By SMOTE method

(Synthetic minority oversampling

method) So [Accuracy and

Change]

2) So [overfit Happen] means

Data Leakage Happens that

why train data highly memorized

So Using validation method like

[Cross-validation] → Stratified k-fold

machine learning models to splitting into

multiple training and validation to

prevent [overfit, underfit, data Leakage]

3) So am. But am exploring a ~~data~~ Accuracy

Using Hyperparameter tuning to find Best parameter

of particular Algorithms using (Grid search CV)

after using → [Hyperparameter tuning may be increasing a Accuracy]

↓

[but not best for (overfit)]

(or)

[To avoid over depth] (or) split Leaves ← [Sometime they help to prevent overfit also]

---

4) Then am trying a (Deep Learning)

Neural network [fully connected layer]

↓

framework as (tensorflow)

addit

adding a two hidden layers →

[1st hidden layer] have [128 Neurons]

&

activation function is (relu)

(rectify linear units) ← keep positive value & remove negative value

it model is not have loss function it run [back propogation] like gradient descent fn. work like (loop) when loss is reduced only loop next step

Droup out layer used (0.3) that mean. 30% neurons will droped when model prevent overfitting like early epoch stoped

activate

[2nd hidden layer] have [64 Neurons]

&

activation function is [relu]

Dropout used as saved (0.3)

[output layer] use activation: "softmax"

optimizer = ["adam"]

↓

→ [learning rate control how much the weight are adjusted in each steps]

* Each neurons in a layer has weight that determine its importance in making prediction

* [The model updated the weight during back propogation]

Learning rate directly affect weight

Learning rate. → Adam ✓
- Adart learning rate dynamically for efficient training
- automatically adjust

→ SGD
- used a fixed learning rate

[Softmax Activation in output layer]

↓

is used for multiclass classification problems

predict multiple categories.

[Back propagation] → uses for loss gradient from Softmax to update weight in all layers (including Relu layers)

Relu → ensure deep learning efficiency]

Softmax → ensure correct probability output]

↳ (avoid to get one class getting very large scores)

↳ [ensure proper probability distribution among all classes]

---

2) (precision & Recall (F1-Score))

↓

based [Confusion Metrix]

↓

of [classification problems]

If [Detected Fraud is task]

↓

1) precession is work actual Fraud & [True positive]

predicted Fraud (actual negative)

↓

[false positive]

wrongly

so. actual negative ~~falsely~~

predicted as predicted positive

like as [false positive]

↓

* It is a good thing because

because it detect fraud from

genuine clim.

But. false positive count increase means its false alarm. it create investigative ~~again~~ stops may genuine claim.

may or genuine claim &

recall → actual Fraud ~~positive~~ & predicted heruire

(actual Fraud)

(false negative)

It is a good thinks sometimes

Some insurence claims may stops because doubts of fraud so. it move on heruire mean.

insurence claims will happen & customer satisfaction.

So in this dataset.

should maintain Recall is higher than precision

F1 score is calculated of precision & Recall

balance (or) imbalance

So need to [F1-Score]

if 1) task is only [find fraud] mean

concentrated of precession

Especially → [false positive]

---

2) if the task is only [find heavenire] mean

concentrated of recall

especially → [false Negative].

---

3) [performance matrixs]

↳ [error based matrixs]

1) [Mean square Error] ↳ [show error bigger]

↳ ~~Actual~~ Error was squared

↳

2) [Root mean square Error]

↳ is Error orginal values

MSE calculated

↳ distance of (predict b/w actual points)

Accuracy is above 9...

been generalized.

well mean

but [MSE very high]

it affected prediction

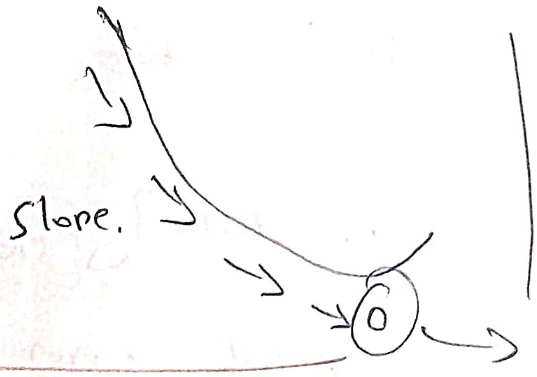
↓

To reduce [MSE] use [Gradient descent]

act to [changing weight to find slope]

to reduce error.

[Global ~~minimum~~] in [U-shape] ↓



[Loss function Curve]

the lowest point is mean by [Global minimum]

[that weight will be used the model]

~~Y~~ → Loss (Error)

X → model parameters (weights & biases)

↳ optimal learning rate [find best]

to avoid overshooting

* The U-shape represents how Loss decrease as we train a model

* Gradient descent follows this Curve to find best optimal weight.

* A small learning rate ensures smooth convergence, avoid overshooting.

---

when use [Regularization]

↓

for 1) High variance overfitting happens

↳ [best fit like connected all points]

2) unstable weight update

3) large or unnecessary weight values

4) Too many feature complexity

[Regularization helps stabilize training and improve generalization]

2) L1 (Lasso) Regularization

* (Shrinks some weights exact to zero)
  performing feature selection)

* Help you when you have irrelevant features.

* Encourages a sparse model (simpler, less complex)

1) focusing important features
   remove irrelevant

2) smaller & faster easier to interpret

3) only few feature contribute to prediction

3) L_2 (Ridge) Regularization

↳ shrinks weights closer to zero (not exactly zero)

* Reduce model complexity while keeping all features

* Helps prevent overfitting and improve stability in training

↳ reduce overfitting without dropping feature.

Elastic net → use $L1 + L_2$

↳ (L_1) some feature should be eliminated

↳ some should be reduced (L_2)

1) Try a smaller learning rate (lr = 0.0001 or 0.0005

2) use L_2 (ridge)

1) Sparse vs Dense models

(Sparse → L_1 (lasso), Dense → L_2 (Ridge))

Feature	Sparse model	Dense model
Number of Non-zero weights	Few (many weights are 0) zero	Many (most weights are non-zero)
Computational Efficiency	Faster, smaller	slower, Heavier
Feature selection	select important Features	used all features even unimportant ones
Memory usage	less memory (good for deployment)	More memory
eg: Regularization	L_1 (Lasso) Regularization	L_2 (Ridge) Regularization

* AUC - ROC Score

↓

Area under curve & Receiver operating characteristic

evaluating fraud detection model's ability to distinguish between fraudulent and genuine claims

---

A good model ROC - curve

↓ closer to top left :

1) It detected fraud well (high TPR)
2) It makes few false alarms (low FPR)

Scores:

↳ 0.5 → Random guessing (bad model)

0.6 - 0.7 → poor better than random

0.7 - 0.8 → Decent fraud detection

0.8 - 0.9 → Good fraud detection

0.9 - 1.0 → Excellent fraud detection

Threshold for AUC - ROC Curve

↓

Youden's J statistics

↓

J = TPR - FPR

Youden's J statistics

① ↳ for Balanced dataset (ROC Based)

F1 - Score method

② ↳ for imbalance dataset

↓

(fraud cases are rare)

F1 Score = 2 * (precession * recall) / (precession + recall)

↓

precession = recall

Cost - Base - method

↳ false positive / false negative

different costs

↳ false fraud alert are expensive

AUC - ROC

↓

1) can Handles → Imbalanced well

↳

2) Measure how well the model separates fraud from Genuine Cases

3) work for both balanced & imbalanced dataset

(5)

(x) Clustering models //

↓

Customer Segmentation

1) silhouette score

↓

how well clusters are formed & seperated

↓

(2) using →

1) k-means clustering / Hirarchical clustering

2) DB Scan clustering

↓

(3)

to find clusters → best count of clusters

1) Elbow Point for number of clusters in k-means

2) Dendogram is used for Hierarchical clustering

3) Find best (eps & radius) for DBSCan

Silhouette score mean by how clusters are seperated and values are aligned on particular clusters is correctly then cluster number are values are correctly assigned or not.

cluster ~~scores~~ find based on

1) it using k-mean clustering

using Elbow chart to find the number of cluster best for this dataset

2) it using ~~Dend~~ Hirarichal clustering

to using Dendrogram chart to find the number of clusters best to find (lopest tired endely connect)

best cluster

3) using DB Scan → (epsilon radius)

(eps) → no of values ~~bas~~ com based on Particular radius

Davies - Bouldin

↳ [Concentrate less number of clusters]

(or)

[fewer clusters]

To avoid Too many clusters

Causing overlaps

(Or)

over segmentation

NLP - Models

1) Sentiment Analysis → Accary

2) Translation → BLEU Score

Accuracy & fluency

3) Text summarization → ROUGE Score

based on Actual output vs model Generated output

1) Sentiment Analysis

using a Real time dataset

Twitter Data → 65,000 data.

Check missing values & Duplicates

take only - positive, negative & neutral.

1st step

→ Text preprocessing

* Stop words Removal
* punctuation & special character removal
* Lowercasing, POS.
* lemmatization

2nd Step

Vectorization

↓

1) TF-IDF

2) word 2 vec

3) BERT Embedding

✓ 1) [TF- IDF] → (Statistical method)

work based of the

give number of the Tokenized word

based on frequency (or) give number

based on importance of the word

Highly Cause of Dimensionality

✓ 2) [word 2vec] → neural network based

> It reduce High dimensional space

multiple dimension in single column

Bag of words → Like Grams

3) BERT (Embedding)

↓

[Bidirectional Encoding Representations from Transformers embedding.]

↓

Its pretrained Model.

because of English words already trained so run our own model pic the words and ~~pretrain~~ matching the words into pretrained matchings.

Developed by Google

Then run into machine learning Algorithm.

BERT produce 768 Dimension in one columns so easy to use SVD Truncad to reduce combained into 50D or 1D

SOD is pick to all information

But ID is faster and lighter

But may miss the Important Information

Accuracy get → 75% on Random Forest

{

But prediction work perfectly }

---

2) Translation :

All the preprocessing steps

Remains same for the sentiment Analysis

sentence transformer

1) Similarity check → cosine_similarity,

↓

I am using English to French

similarity check → [English correctly translated to French]

that is similarity check //

2) Step 2

Check BPE (Byte per Encoding)

↓

used in

① GPT-2, Marian MT ...

② Merge Frequent characters

③ work well multiple with Language

④ Smaller & faster

---

3) ~~Ab~~ Step -3

Fine Tuning

- 1) MBART (Multilingual BART)
- 2) MT5 (Multilingual T5)
- 3) M2M100 (facebook's many to many)

All is [Failure computation high Expenses]

& prediction Failure Because of poor/low Data

4) ~~Step-3~~

4) Step-4

↓

so now we pretraind data

without Fine Tunning

No suite for :- Insurance, Medical,

and other confidential project.

we [Only own Data Trained

with fine tunning]

prediction succees

3) Text summarization

↓ [create a real fine Dataset

of from kagele

News Article based]

Text & Headlines

All the preprocessing steps Remain Same

use Autotokenizer from BERT Embedding

① for 2 columns

② then fine tuning for cleaned Raw Data

I am using Bart-small

else others are computational high

so using ~~true~~ this but train only

Allow very low datasets

Rouge score : 0.43

only Rouge 1 & Rouge L

prediction success

2) Business Impact metrics

Fraud Detection & prevention

↓

using Isolation forest & Auto encoder

1) Isola [used contamination rate based on fraud Detection]

if client need to [capture All clients]

isolation forest is best capture all clients based on contamination rate (or)

capture all outliers and labeled as fraud even (low, medium) risk also

(or) client ask improve customers so

capture only extreme frauds (or)

(high Risk frauds)

Best Auto encoder

↳ capture high Risk fraud only

(or)

Extreme outliers

so it reduce False Alarms //

Fraudent payout → Reduction (↓)

Based on Filed claims

↓

1) after the Insurence warant finish next day / month claim will happen

2) Day since Issuence (policy date - claim Date)

3) claim Income Ratio above 20%

↳ (Annual Income - claim Income)

False claims Detection rate (1.)

↓

Based on **Precession**

↓

Concentrate **False positive**

↓

Predicted fraud

↓

from Actual genuine

+

[Actual fraud] →

[True positive]

Try to Reduce False positive

its ⇒ [False Alarm] when increase

False positive

↳ Its increase company

1)

expenses

⇒

2) And stop claiming

genuine claims also.

Increase in correct policy price (or)

↓

Correct policy find

↓

1) Based on ~~dob~~ zero claim history

2) claim Income Ratio lower.

3) Days since Issued high.

4) Descent premium amount payer

) These have claim ~~Amount~~ high

---

else Frauds

↓

policy price was Increased

based on middle Age or old Age

↓

fraud Mostly Happens

1) Health & Auto

policy price Increased

Customer Segmentation

based on

- Claim frequency
  &
  policy upgrades ~~upgraded~~

&

Age & Income

---

1) used as a k-means clustering, Hierarchical & DBSCAN clustering

---

Customer Retention Rate (%)

1) (long time policy ~~order~~ holder)

&

(policy upgrades)

(3) [Operation Efficiency]

{

[Claim processing Efficency]

{ using [Chatbot for]

Q & A

Type 11

Reduce [Human Interference]

[Maximum Avoid to

Human Interference

to know About

Policy Details]

↓

[BERT

+

FAISS]

Fact Based Answering from

Structured Data

[Convert Columns into BERT]

↓

Then BERT Embedding Store into

[FAISS Index] for fast retrieval