Logistic Regression

- Logistic Regression
  - Don't be confused by the use of the term "regression" in its name.
- Logistic Regression is a classification algorithm designed to predict categorical target labels.

Section Overview
- Transforming linear Regression to logistic regression.
- Mathamaticall theory behind Logistic Regression
- Simple implementation of Logistic Regression for classification problems.

- Interpreting Results :-
  - Odds Ratio and coeficients
  - Classification Metrics
    - Accuracy
    - Precision
    - Recall
    - ROC curves (Receiver Operator characters curves) ✓
- Multiclass classification with logistic Regression.
- logistic Regression project
- logistic Regression project solution.

- Logistic Regression will allow us to predict a categorical label based on historical features data -
  - The categorical target column is two or more discrete class labels.

- Classification algorithm predict a class or categorical label:
  - class 0 : Car Image
  - Class 1 : Street Image
  - class 2 : Bridge Image

keep in mind, any continuous target can be converted into categorical through discretization.
- Class 0 price $0 - 100K
- class 1 price $100K - 200K
- class 2 price $200K

- Classification algorithm also often produce a probability prediction of belonging to a class:
  - Class 0: 10% probability - Car Image
  - Class 1: 85% probability - street Image
  - Class 2: 5% probability - Bridge Image

  - model reports back prediction of class 1, image is a street.

- Also note our prediction $\hat{y}$ will be a category, meaning we would be able to calculate a difference based on $y - \hat{y}$.

• car Images - street Images does not make sense.

- We need to discover a completely different set of error metrics and performance evaluation.

3- Logistic Regression

- Theory & Intuition part One

Logistic Regression works by transforming a linear Regression into a classification model through the use of logistic function

$$f(x) = \frac{1}{1+e^{-x}}$$

• Logistic function = Sigmoid function

• for now we just are discussing a logistic function not logistic Regression. Its theory and past developement.

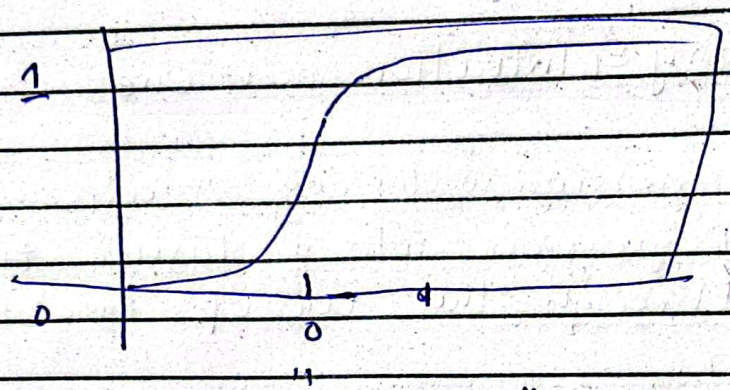
- 1830 - 1850 - Pierre Francoise Verhulst developed logistic function:

$$f(x) = \frac{1}{1+e^{-x}}$$

In 1883 : Logistic Regression function was independently developed in field chemistry as a model of autocatalysed by Wilhelm Ostwald.

- 1830-1850 - Developed for the purposes of modelling population growth.

- Why we need for a logistic functras versus a logarithmic function?



Note: There is a family" of logistic Regression.

- $\text{erf}\left(\frac{\sqrt{\pi}}{2}x\right)$
- $\tanh(x)$
- $\frac{2}{\pi}\text{gd}\left(\frac{\pi}{2}x\right)$
- $\frac{x}{\sqrt{1+x^2}}$
- $\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)$
- $\frac{x}{1+|x|}$

- Notice the "leveling off" behaviour of the curve.

output range
0 – 1

- Many natural real world systems have a "carrying capacity" or natural limiting factor.

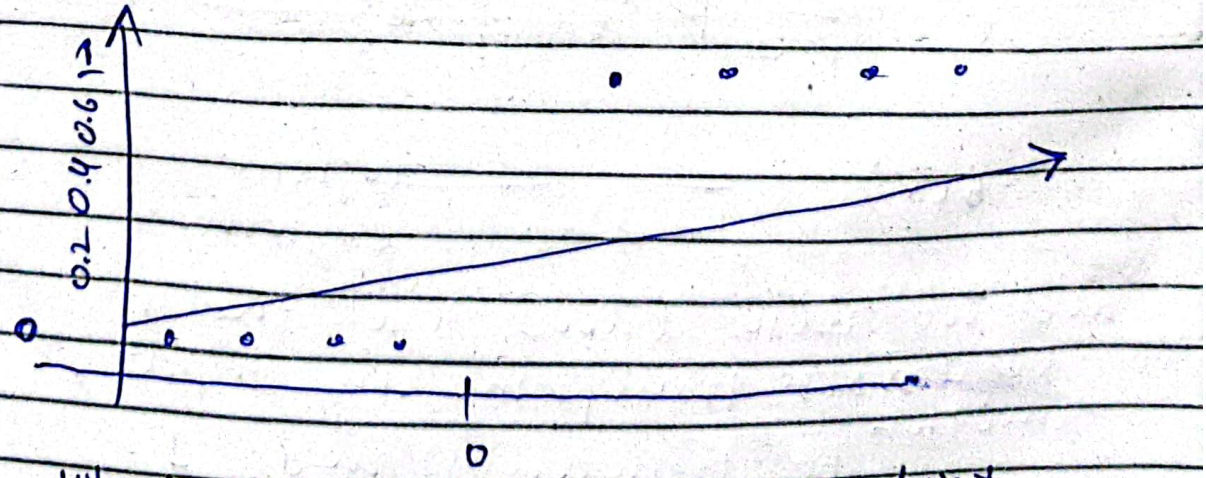
- 1940s : Using logistic function for statistical modelling was developed by Joseph Berkson.

- 1944 : "Application of the logistic function to bio-assay" in the journel of the American Statistical Association.

004 - Logistic Regression. Theory & Intuition
part 2 - Linear to Logistic

- our dataset

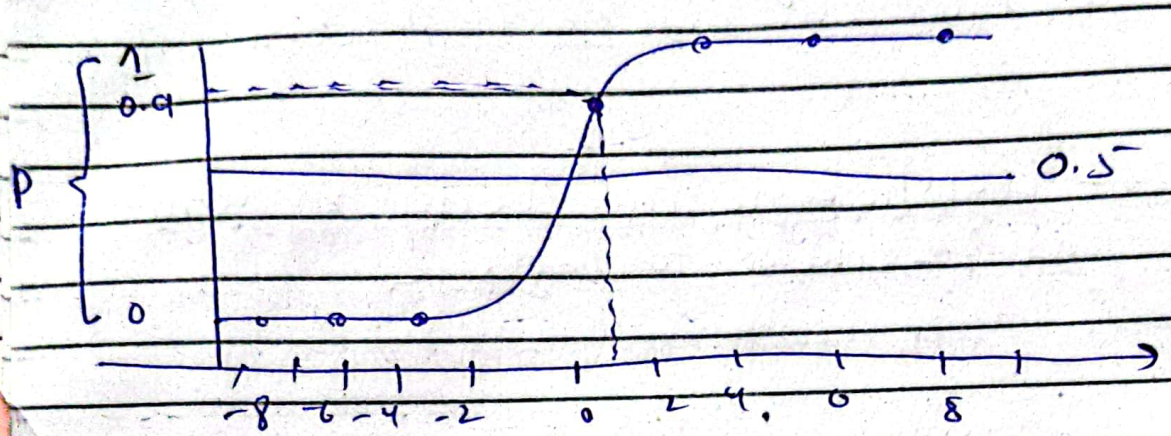
Income	loan paid
-5	0
-4	0
-2	0
-1	0
0	0
2	1
3	1
4	1
5	1
7	1



What if we want to predict if a person would be able to

- Sonenie's income?
- fitting the Linear Regression would not work (recalls Anscombe's quartet).

- we could make use of a logistic function



the above point will be consider 1 and assumed to have 90% probability based on direction.

mx,

005 Logistic Regression - Theory & Intuition part two Linear to logistic Math

$$\hat{y} = \beta_0 x_0 + \cdots + \beta_n x_n$$

$$\hat{y} = \sum_{i=0}^{n} \beta_i x_i$$ — linear Regression

$$\sigma(x) = \frac{1}{1+e^{-x}}$$ — logistic Regression

and we also know that the logistic Regression transforms any input to function be between 0 and 1.

$$\hat{y} = \sigma[\beta_0 x_0 + \cdots + \beta_n x_n]$$

$$\hat{y} = \sigma\left(\sum_{i=0}^{n} \beta_i x_i\right)$$

$$\hat{y} = \frac{1}{1 + e^{-\sum_{i=0}^{n} \beta_i x_i}}$$

- First we need to understand the terms odds.
- A term you may be familiar with from gambling odds.
- In gambling odds are often referred to in the sense of N to 1.
- But where does this actually come from?

$$\frac{p}{1-p} \quad \rightarrow \text{probability of happening / not happening}$$

- Solving for logs odds

$$\hat{y} = \frac{1}{1 + e^{-\sum_{i=0}^{n} \beta_i x_i}}$$

$$\hat{y} + \hat{y} e^{-\sum_{i=0}^{n} \beta_i x_i} = 1$$

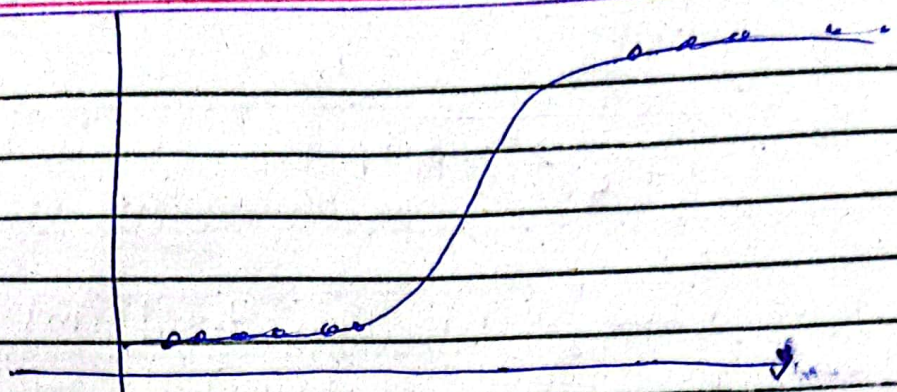
$$\hat{y} \cdot e^{-\sum_{i=0}^{n} \beta_i x_i} = 1 - \hat{y}$$

$$\frac{\hat{y}}{1-\hat{y}} = e^{\sum_{i=0}^{n} \beta_i x_i}$$

$$\ln\left(\frac{\hat{y}}{1-\hat{y}}\right) = \sum_{i=0}^{n} \beta_i x_i$$

What would the function come look like in terms of log-odds?

$$\ln\left(\frac{0.5}{1-0.5}\right)=0$$

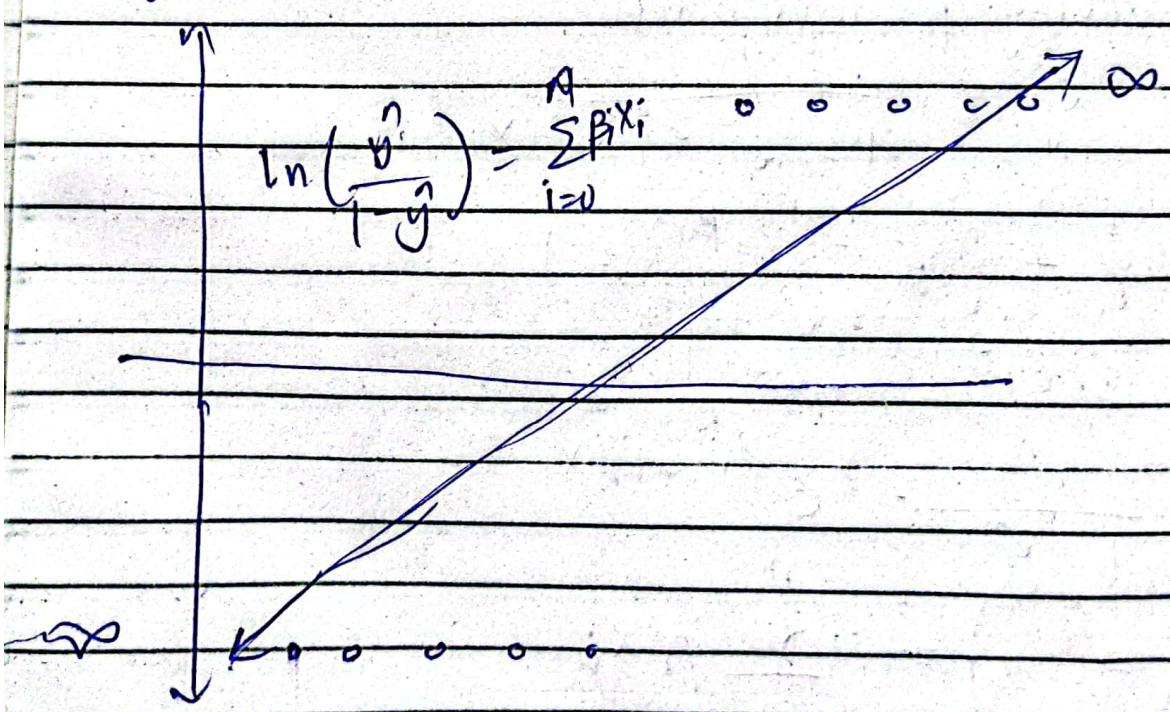


- As p goes to 1 then log odds become ∞

$$\lim_{p\to 1} \ln\left(\frac{p}{1-p}\right) = \infty$$

- As p goes to 0 then log odds become $-\infty$

$$\lim_{p\to 0} \ln\left(\frac{p}{1-p}\right) = -\infty$$



- Since the log odds scale is non-linear a β value can not be directly linked to "one unit increase" as it could in linear Regression -

$$\ln\left(\frac{y}{1-y}\right) = \sum_{i=0}^{n} \beta_i x_i$$

- Sign of beffiuent
  - possitive β indicate an increase in likelihood of belonging to 1 class with increase in associated X features.
  - Negative β indicates an decrease in likelihood of belonging to 1 class with increase in associated X features.

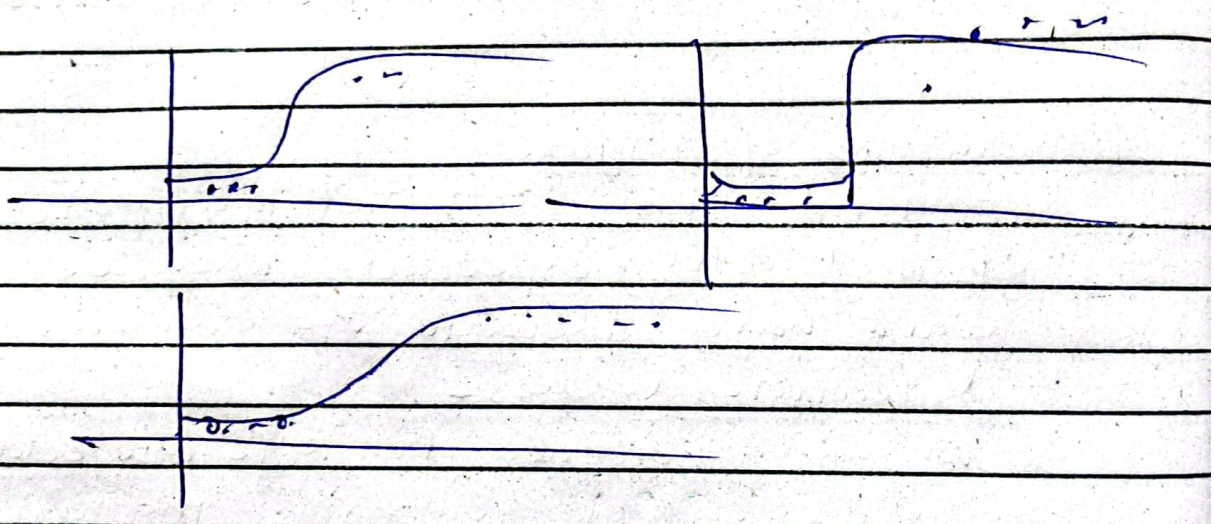
- Magnitude of coeffiuent
  - Harder to directly interpret magnitude of β directly, especially when we could have discreate and continous X features values.
  - We can however begin to use odds ratio, essentially comparing magnitude against each other.
  - Comparing magnitudes of coefficients against each other can lead to insight over which features have the strongest effect on prediction output.
- We will discuss the basic of fitting the best curve with maximum likelihood in the next lecture.

106 Logistic Regression - Theory - Intuition

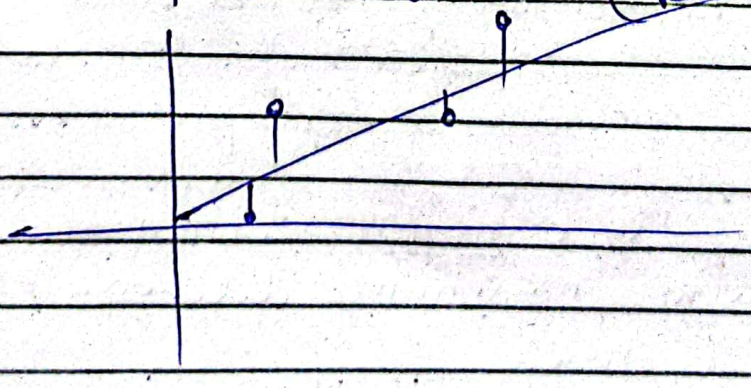
part three : finding the best fit.

- Logistic Regression uses Maximum likelihood to find the best fitting model.
- This lecture will also display cost function and gradient descent that is solved for by the computer.

Here we see three different β values. How do we measures which is best fit?

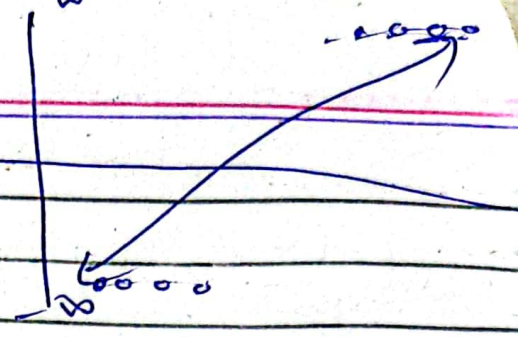


Recall in linear Regression we seek to minimize the Residual sum of squares (RSS).



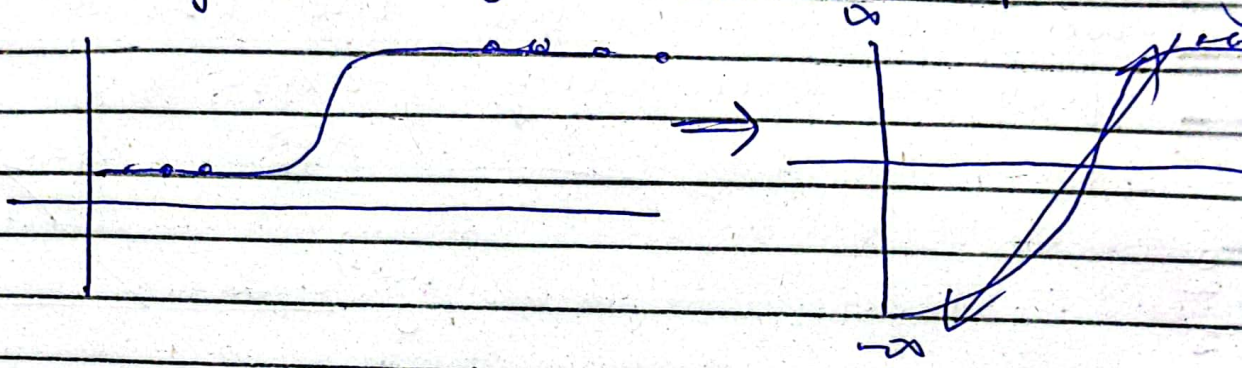
Unfortunately, ~~the~~ log odds targets are at infinity, making RSS unfeasable.

$$\ln\left(\frac{\hat{y}}{1-\hat{y}}\right) = \sum_{i=0}^{n} \beta_i x_i$$



Maximum likelihood:-

- The first step is to know that we can go from log odds back to probabilities



$$\ln\left(\frac{p}{1-p}\right) = \ln(\text{odds})$$

$$\frac{p}{1-p} = e^{\ln(\text{odds})}$$

$$p = (1-p)e^{\ln(\text{odds})}$$

$$p = e^{\ln(\text{odds})} - pe^{\ln(\text{odds})}$$

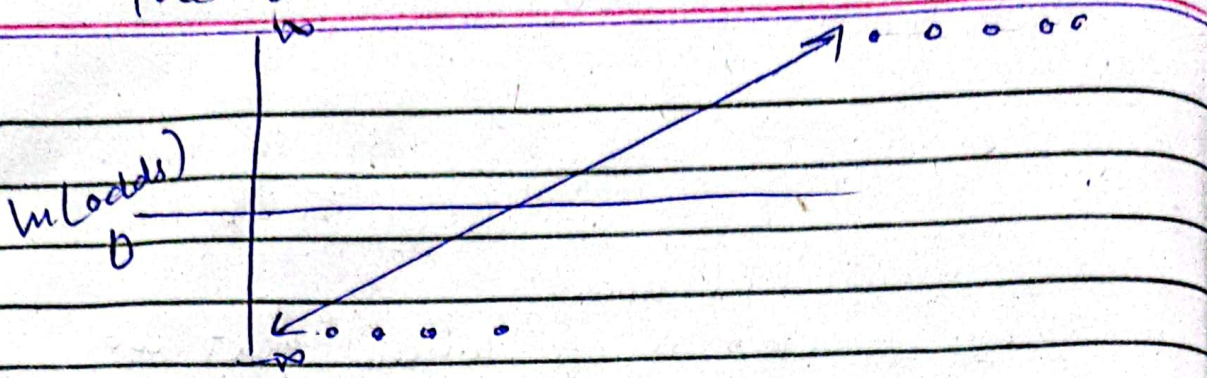
$$p + pe^{\ln(\text{odds})} = e^{\ln(\text{odds})}$$

$$p\left(1 + e^{\ln(\text{odds})}\right) = e^{\ln(\text{odds})}$$

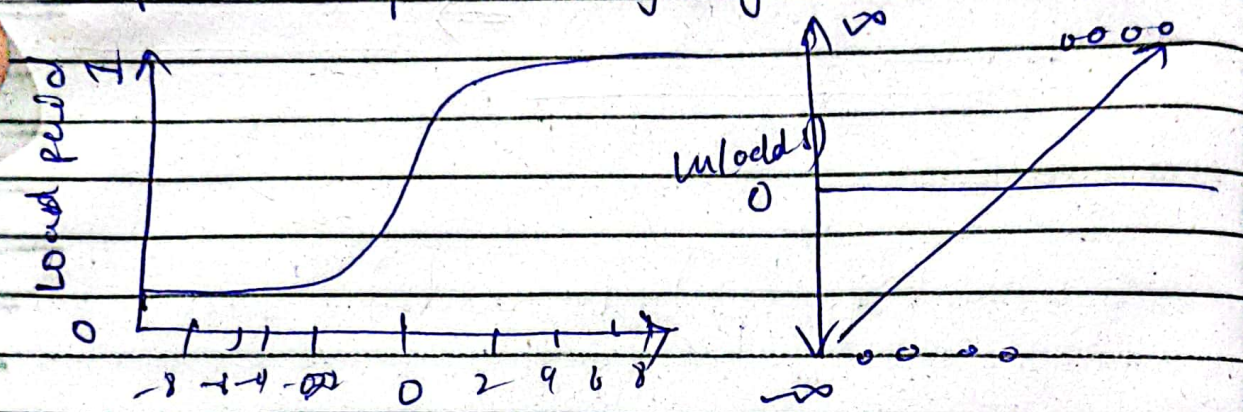
$$p = \frac{e^{\ln(\text{odds})}}{1 + e^{\ln(\text{odds})}}$$

Now we are able to convert $\ln(\text{odds})$ into probability.

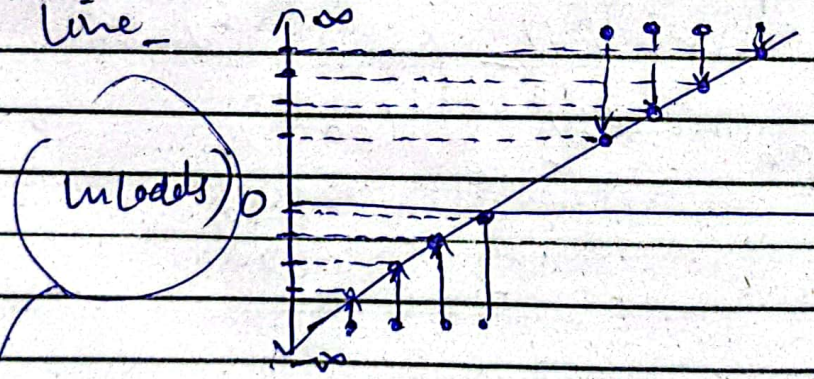
- We choose a line in the log(odds) axis and project the point on the line



- we also know that line has a form in probability y-axis-

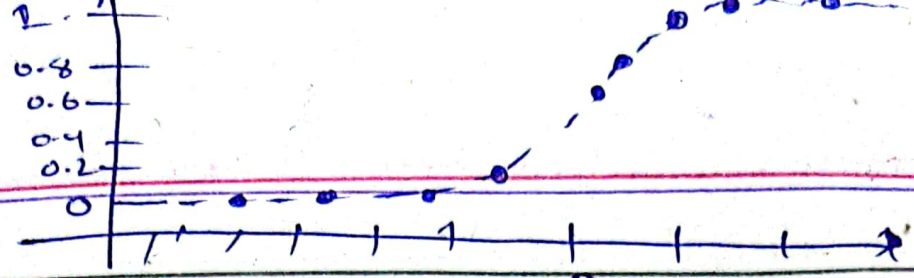


- we chose a line in the log(odds) axis and project the points on the line-



- calculate the logs odds for the projected points on the this Line put that project points

$$p = \frac{e^{\ln(\text{odds})}}{(1 + e^{\ln(\text{odds})})}$$



- plot these values as the probabilities on the Logistic Regression model.
- And we now measure the likelihood of these probabilities.

→ likelihood = product of probabilities of belonging to class 1.

Note
- if it has a perfect fit
- blue points should be all the way at 1
- Black should be at 0

likelihood = 0.9 x 0.8 x 0.65 x 0.55 x (1-0.2) x (1-0.08) x (1-0.02)

(1-0.2) ... : class 0

likelihood = 0.129

↓

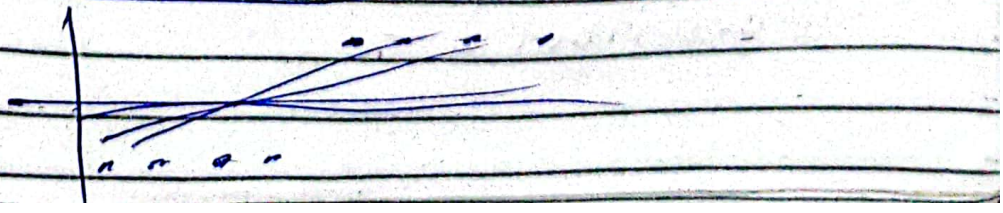
we want to maximize this to 1

Note:
In pratical we actually maximize the log of the likelihood
(e.g. $\ln(0.9) \times \ln(0.8) \times \dots$)

↓

Mathematics works better

- We just there is some set of coefficient that will maximize the likelihood log.



- While we are trying to maximize the likelihood, we still need something to minimize, since the computer is gradient descent methods can only search for minimums.
- In terms of cost function, we seek to minimize the following (logloss):

$$J(x) = -\frac{1}{m}\sum_{j=1}^{m} y^j \log(\hat{y}^j) + (1-y^j)\log(1-\hat{y}^j)$$

$$J^j(x) = -\frac{1}{m}\sum_{j=1}^{m}\left(y^j \log\left(\frac{1}{1+e^{-\sum_{\phi=0}^{n}\beta_i x_i^j}}\right) + (1-y^j)\log\left(1-\frac{1}{1+e^{-\sum_{i=0}^{n}\beta_i y_i^j}}\right)\right)$$

- Just with Linear Regression gradient decent can solve this for us!

007 LogisticRegression with Scikit-learn

~~part one: Exploratory Data Analysis~~

```
# code
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("heamy_test.csv")
df.head()
df.describe() -
```

df['test_result'].value_counts()

out 1 3000
0 2000

sns.countplot(df["test_result"])

sns.boxplot(x="test_result", y="age")
sns.boxplot(x="test_result", y="physical_s")

sns.scatterplot(x='age', y="physical_score", data=df, hue="test_result")

↳ will tell us about intuition

sns.pairplot(hue="test_result")
sns.heatmap(df.corr(), annot=True)

sns.scatterplot(x="physical_score", y="test_res

from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax = scatter(df['age'], df['physical_score'], df["test_result"], c='test_result')

008 Logistic Regression with Scikit Learn

part two : creating and Training a Model

```
# code
X = df.drop("test result", axis=1)
y = df['test result']

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

scaler = StandardScaler()
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)

from sklearn.linear_model import LogisticRegression

help(LogisticRegression)

log_model = LogisticRegression()
log_model.fit(scaled_X_train, y_train)

log_model.coef_

y_pred = log_model.predict(scaled_X_test)
log_model.predict_proba()
```

009 Classification Metrics - Confusion Matrix and Accuracy

part 1: Confusion Matrix Basics

- You've probably heard of terms such as "false positive" or "false negative". As well as metrics like "accuracy".
- But what does these terms actually mean mathematically?

- Imagine we have developed a test or model to detect presence of a virus infection in a person based on some biological feature
- we could treat this as a logistic Regression predicting:
  - 0 - Not Infected (Test Negative)
  - 1 - Infected (Test positive)

- It is unlikely our model will perform perfectly. This mean there 4 possible outcomes:
  - Infected person test positive
  - Healthy person test negative
  - Infected person test negative
  - Healthy person test positive

- Based on these 4 posibilites, there are are many error metrics we can calculate..
- First let start by visualizing these four possibilities as a matrix-

predicted	Actual Infected	Healthy
Infected	True positive	fake positive
healthy	fake negative	True negative

100 people - 5 are infected

pred	Infected	Healthy
Infected	4	2
Healthy	1	93

Accuracy :-

How often is the model correct ?

Acc = (TP + TN) / Total

$= \frac{(4+93)}{102} = 97\%$ Accuracy

- Is this good value ?

The accuracy paradox could be caused by imbalance Label data

* This is the Accuracy paradox!
  - Any classifier dealing with Imbalanced classes has to confront the issue of the accuracy paradox.
  - Imbalanced classe will always result in a distorted accuracy reflecting better performance than what it is truely warranted.

* Imbalance classes are often found in real world datasets:
  - medical condition can affect small portion of the population.
  - Fraud is not common (e.g. Real vs Fraud credit card Usage).

- If a class is only a small percentage (n%), then a classifier that always predict the major class will always have an accuracy of (1-n).
  - In our previous case examples we saw infected were only 5% of the data.
  - Allowing the accuracy to be 95%.

- This mean we should rely on accuracy.

10 - Classification performance Metrics

part 2 : Precision and Recall

Lets explore three more ~~classics~~ metrics that can help give a clearer picture of performance :

- Recall (a.k.a sensitivity)
- precision (~~a.k.a sensitivity~~)
- F1-Score

Let begin with Recall

Recall :-

When it is actually is a positive cas, how often is it correct ?

$$\frac{(TP)}{\text{Total actual positive}} = \text{Recall}$$

$$\text{Recall} = \frac{4}{5} = 80\%$$

If always positive

$$\text{Recall} = \frac{0}{5} = 0\%$$

$$\frac{TP}{TP+FN} = \text{Recall}$$

Now lets explore precision:

precision:-
When prediction is positive, how often it is correct.

$$\frac{TP}{TP+FP} = \text{precision}$$

Total predicted positive

$$\frac{TP}{TP+FP} = \text{precision}$$

$$\frac{4}{6} = 0.666\ \%$$

If always healthy

$$\frac{0}{0+0} = 0$$

- Recall and precision can help illuminate our performance specifically in regards to the relevant or positive cases.
- Depending on the model, there is typically a trade-off between precision and recall, which will I will explore later on with the ROC curve -

ROC (Receiver operator characteristic curve)

F1-Score :-

Since precision and recall are related to each other through the numerator (TP), we often also report the F1-score, which is harmonic mean of precision and recall.

$$F = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

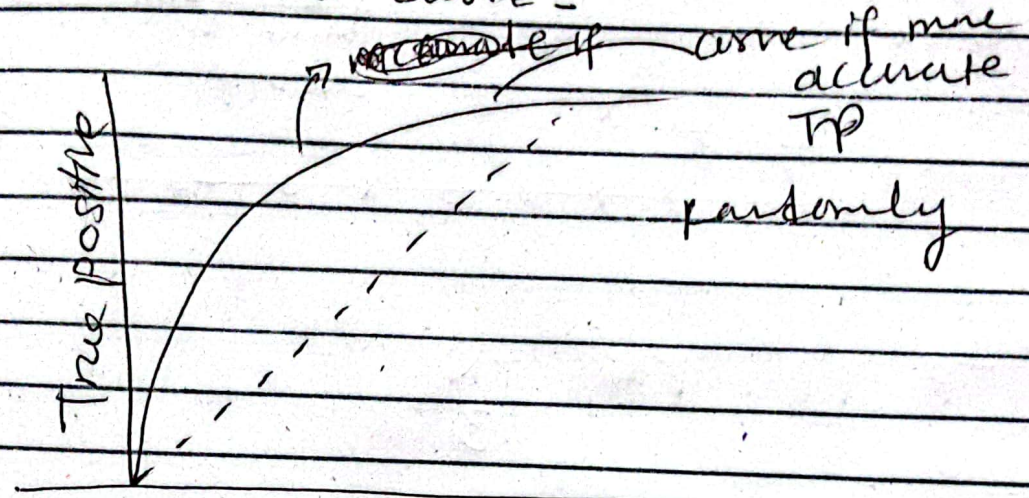
- The harmonic mean (instead of the normal mean) allows the entire harmonic mean to go to zero either precision or recalls ends up being zero.

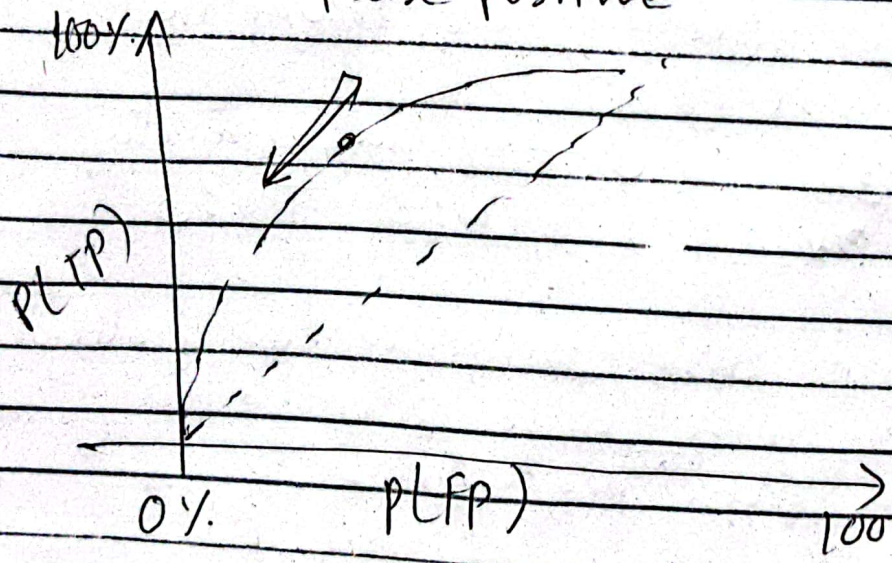
As a final note there are many more metrics available : can check them online.

041 Classification Performance Metrics part 3: ROC curves

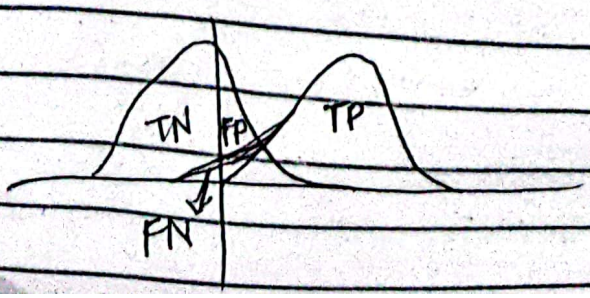
History :-

During WW2 Radar was developed new to evaluate radar performance they developed Receiver operator ~~Curve~~ Characteristic curve -





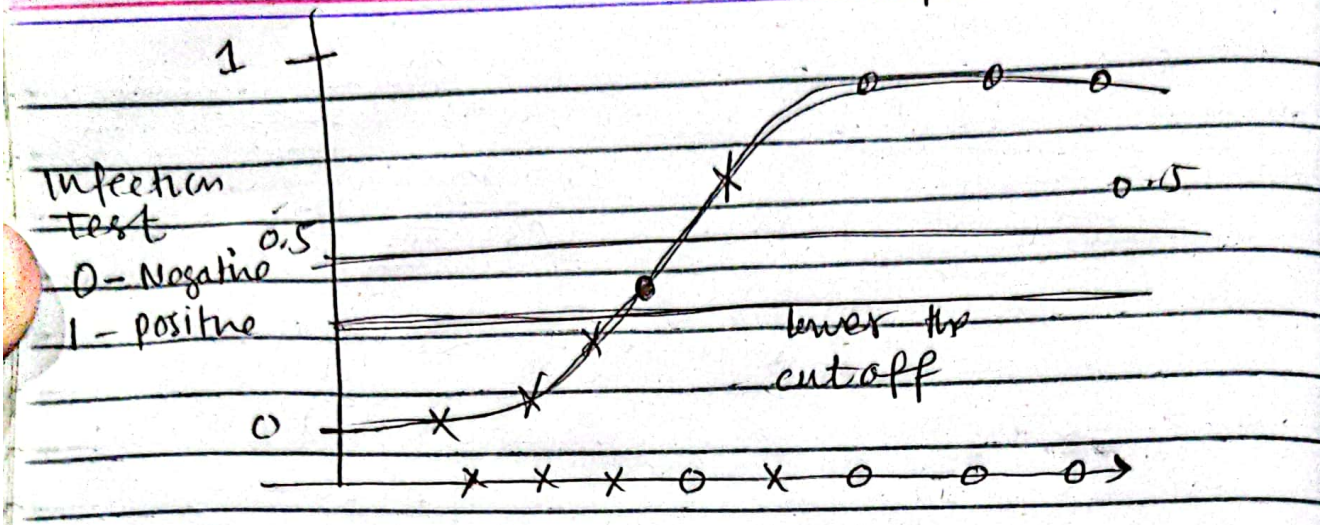
There can be a trade off between True positives and false positives -



TP	FP
FN	TN

Our primary test: infection

Actual
X Negative
O positive



TP	FP	FN	TN
3	1	1	3

But sometime we lower the cutoff.

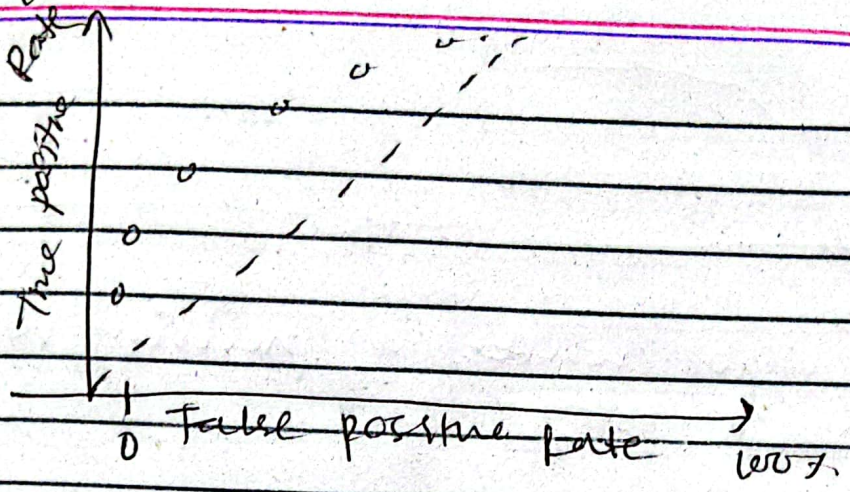
3 O — my understanding

TP: ~~3~~2 FP: 2, FN: 0 TN: 3

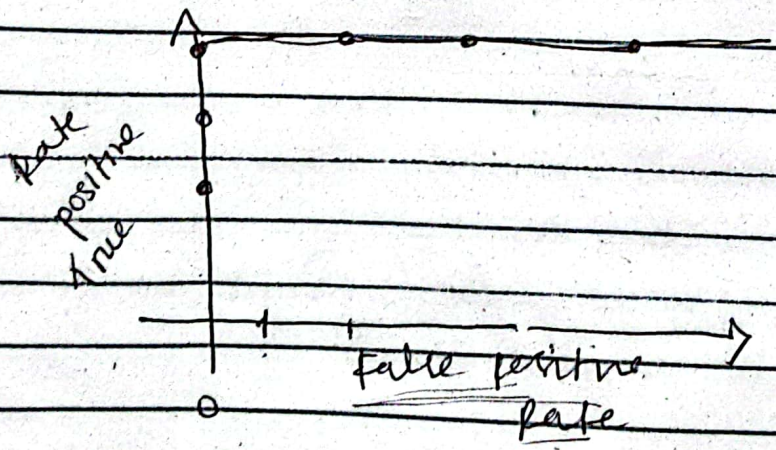
In certain cases/situation we gladly accept more false positive to reduce false negatives.

- Imagine a dangerous virus test we ~~store~~ would much rather produce false positive too and later do more stringent examination then accidently release a false negative!

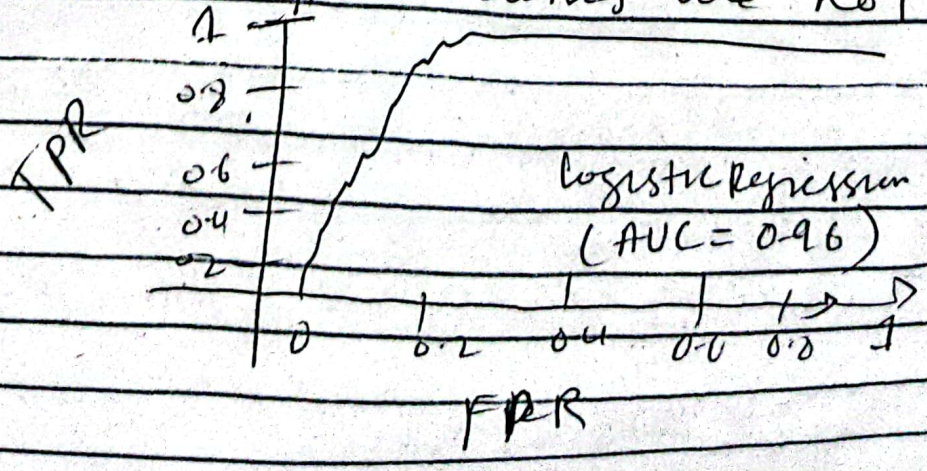
- Chart the True vs. False positives for various cut-offs for the ROC (100% Curve.



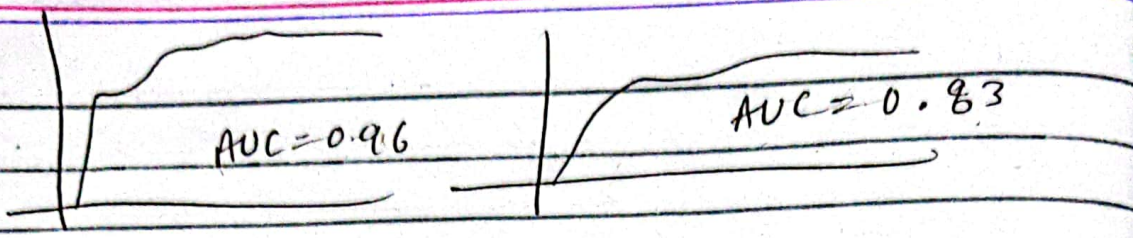
- A perfect model would have a zero FPR.
- Random guessing is the Red line.



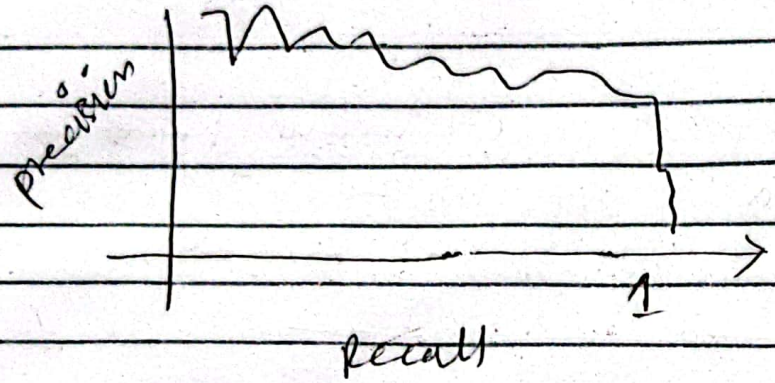
- Realistically with smaller datasets the ROC curves are not as smooth.



- AUC - Area Under the Curve, allows us to compare ROCs for different models.



can also create a precision versus recall curves:



12 Logistic Regression with Scikit-learn part 3 - performance Evaluation.

```
#code
log_model.coef_
df.head()

from sklearn.metrics import accuracy_score,
                            confusion_matrix,
                            classification_Report

y_preds = log_model.predict(X_scaled_test)
accuracy_score(y_test, ypreds)
out 93.0 %
0.93
```

confusion_matrix (y_test, y_preds)
from sklearn.metrics import plot_confusion_matrix
~~plot_confusion_matrix(log_model, scaled_xtest, y_test)~~
normalize = 'all'

len(y_test)

print
(classification_report(y_test, y_preds))

precision_score( )
recall_score( )

from sklearn.metrics import plot_precision_recall_curve, plot_roc_curve

plot_roc_curve(log_model, scaled X_tar, y_test)

~~plot_precision_recall (~~

~~log_model.predict_proba (scaled_X_test)[0]~~

013

Logistic Regression Multiclass Problems

part 1 - Data and Model

Iris Datasets

```
df = pd.read_csv("../csv")
df.head()
df.info()
df.describe()
df["species"].value_counts()
```

sns.countplot(data=df, x="species")
sns.scatterplot(x="petal_length", y="petal width", hue="species")
~~sns.pairplot(data=df)~~
sns.heatmap(data.corr(), annot=True)

X = df.drop('species', axis=1)
y = df["species"]

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.X_test = scaler.transform(X_train)

014) Logistic Regression.
Multiclass problem
part two training and performance Evaluation.

from sklearn.model_selection import GridSearchCV

log_model = LogisticRegression()

hyperparameters:-

penalty: ["l1", "l2", 'elastinet', "none"]
default='l2'

The newton-cg and 'lbfgs' solvers support only 'L2' penalties, "elasticnet" is only supported by 'saga' solver. If 'none' (not supported by the liblinear solve), no regularization is applied

L1_ratio: float if used in elasticnet

multi_class: {'auto', 'ovr', 'multinomial'}
default "auto"
(One versus rest)

solver: {"newton-cg", 'lbfgs', 'liblinear', 'sag', 'saga'}
default = 'lbfgs'
optimization algorithms try to minimize cost function.

= Code

model = LogisticRegression(solver='saga', multi_class='ovr', max_iter=5000)

penalty = ["l1", 'l2', 'elasticnet']
L1_ration = np.linspace(0, 10, 20)
C = np.logspace(0, 10, 20)
param_grids = {"penalty": penalty, "L1_ratio": L1_ratio, 'C': C}

grid_model = GridSearchCV(log_model, paramsgrid = paramsgrids)

grid_model.fit(scaled_X, scaled, ytrain)

from sklearn.metrics import accuracy_score, confusion_matrix

y_pred: grid_model.best_params_

{'C': 11.088

y_preds = grid_model.predict(scaled_X_test)

accuracy_score(y_test, y_pred)

ROC wont work with Multiclass will need to do this using scikit learn code.

//

code

//