Econometrics Project

Classification models comparison on WVS data

Marine Palyan
Vardan Arevshatyan
Sona Asatryan
Haykuhi Danielyan
Liana Isayan

Data & Goal

World Values Survey
Wave 7 (2017-2020)
80 countries
75,956 rows of data

Y **Q57** - Generally speaking, would you say that most people can be trusted or that you need to be very careful in dealing with people?
1 Most people can be trusted
2 Need to be very careful

Our aim is to find **trusting** people which can be useful for different things (selling or convincing something, etc.)

5 classification models applied:

- **Logistic Regression**
- **Decision Tree**
- **Randomforest**
- **KNN**
- **Naive Bayes**

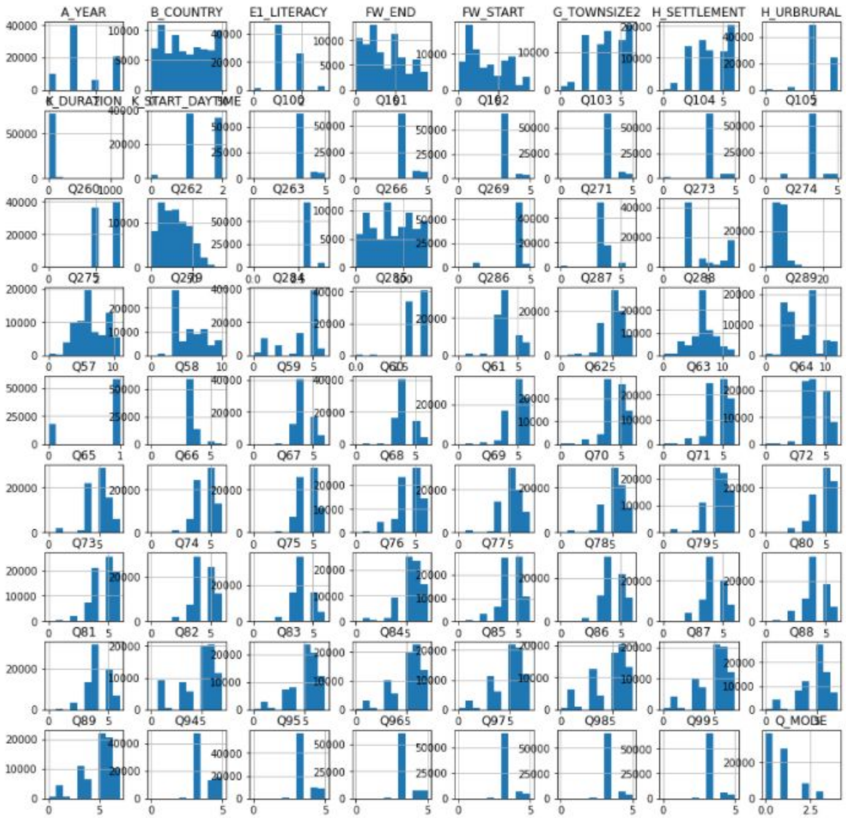
Xs 'COUNTRY', 'YEAR', 'FW_START', 'FW_END', 'DURATION', 'Q_MODE',
'TOWNSIZE', 'SETTLEMENT',
'URBRURAL', 'LITERACY',
'Q58': 'Q63' (trust direction),
'Q64': 'Q105' (trust in),
'Q260' (sex),
'Q262' (age),
‘Q263’ 'Q266','Q269' (residence),
'Q271' (alone?),
'Q273' (marriage),
'Q274' (children),
'Q275' (education),
'Q279' (employed),
'Q284' (public),
'Q285':'Q288' (income),
'Q289' (religiousness)

Descriptive Statistics

```
df1[df1.columns].describe().T
```

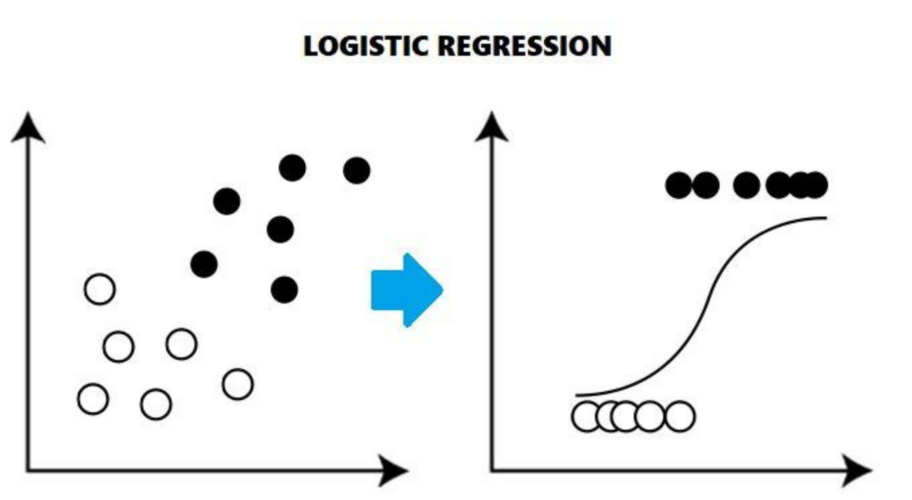
	count	mean	std	min	25%	50%	75%	max
W_WEIGHT	76897.0	1.000003	0.411771	0.100485	0.926213	1.000000	1.000000	19.930071
S018	76897.0	0.661167	0.214587	0.248880	0.483793	0.739098	0.833333	1.000000
pwght	76897.0	3.929099	6.390569	0.006970	0.573330	1.688500	5.380660	32.663040
B_COUNTRY	76897.0	418.747493	252.337219	20.000000	158.000000	398.000000	642.000000	840.000000
A_YEAR	76897.0	2018.485767	1.018354	2017.000000	2018.000000	2018.000000	2020.000000	2020.000000
FW_START	76897.0	201835.097416	96.368544	201701.000000	201801.000000	201807.000000	201910.000000	202010.000000
FW_END	76897.0	201852.280791	98.049076	201703.000000	201805.000000	201810.000000	201912.000000	202010.000000
K_TIME_START	76897.0	10.418434	8.102806	-5.000000	9.060000	12.520000	16.190000	23.580000
K_TIME_END	76897.0	11.536322	8.196057	-5.000000	10.300000	14.000000	17.230000	23.590000
K_DURATION	76897.0	78.991300	419.963883	-5.000000	28.000000	48.000000	65.000000	9999.000000
Q_MODE	76897.0	1.751187	0.865428	1.000000	1.000000	2.000000	2.000000	5.000000
G_TOWNSIZE	76897.0	4.862218	2.996251	-5.000000	3.000000	6.000000	7.000000	8.000000
G_TOWNSIZE2	76897.0	2.810825	2.010257	-5.000000	2.000000	3.000000	4.000000	5.000000
H_SETTLEMENT	76897.0	2.934068	1.872613	-5.000000	2.000000	3.000000	5.000000	5.000000
H_URBRURAL	76897.0	1.189838	0.973675	-5.000000	1.000000	1.000000	2.000000	2.000000
E1_LITERACY	76897.0	-2.094802	2.485881	-5.000000	-4.000000	-4.000000	1.000000	2.000000
Q57	76897.0	1.718494	0.624435	-5.000000	2.000000	2.000000	2.000000	2.000000
Q260	76897.0	1.520631	0.520385	-5.000000	1.000000	2.000000	2.000000	2.000000
Q262	76897.0	42.819759	16.613752	-5.000000	29.000000	41.000000	55.000000	103.000000
Q263	76897.0	1.055425	0.298242	-5.000000	1.000000	1.000000	1.000000	2.000000
Q266	76897.0	466.735789	721.294648	-5.000000	170.000000	400.000000	643.000000	9999.000000
Q269	76897.0	0.715854	1.251328	-5.000000	1.000000	1.000000	1.000000	2.000000
Q271	76897.0	1.256811	0.909220	-5.000000	1.000000	1.000000	2.000000	4.000000
Q273	76897.0	2.599230	2.164190	-5.000000	1.000000	1.000000	5.000000	6.000000
Q274	76897.0	1.700977	1.806812	-5.000000	0.000000	2.000000	3.000000	24.000000
Q275	76897.0	3.459238	2.080049	-5.000000	2.000000	3.000000	5.000000	8.000000
Q279	76897.0	3.072669	2.120695	-5.000000	1.000000	3.000000	5.000000	8.000000
Q284	76897.0	0.673641	2.094917	-5.000000	-1.000000	2.000000	2.000000	3.000000
Q285	76897.0	1.465701	0.801546	-5.000000	1.000000	2.000000	2.000000	2.000000
Q286	76897.0	1.932949	1.090037	-5.000000	1.000000	2.000000	2.000000	4.000000
Q287	76897.0	3.173323	1.212150	-5.000000	3.000000	3.000000	4.000000	5.000000
Q288	76897.0	4.654018	2.277837	-5.000000	3.000000	5.000000	6.000000	10.000000
Q289	76897.0	2.971858	2.694555	-5.000000	1.000000	2.000000	5.000000	9.000000

```
df.hist(figsize=(14,14))
plt.show()
```



Logistic Regression

Logistic Regression



- ❖ Logistic regression is a supervised learning algorithm used to predict a dependent categorical target variable.
- ❖ May be used when predicting whether bank customers are likely to default on their loans.
- ❖ Logistic regression is a classification model, unlike linear regression.

Types of Logistic Regression Models

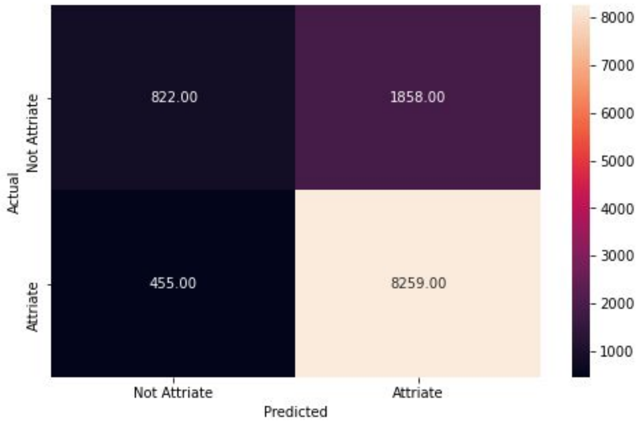
Types of Logistic Regression Models

	Binomial Logistic Regression	Multinomial Logistic Regression	Ordinal Logistic Regression
Number of Categories for Response Variable	2	3 or more	3 or more
Does Order of Categories Matter?	No	No	Yes

- Binary logistic regression
- Multinomial logistic regression
- Ordinal logistic regression

Logistic Regression | Confusion Matrix and classification report

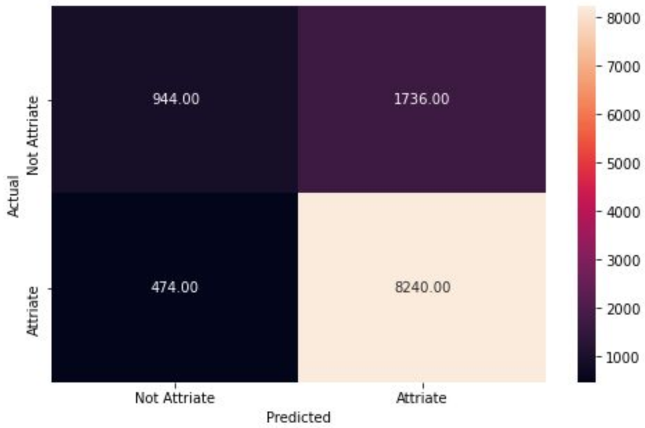
	precision	recall	f1-score	support
1	0.64	0.31	0.42	2680
2	0.82	0.95	0.88	8714
accuracy			0.80	11394
macro avg	0.73	0.63	0.65	11394
weighted avg	0.78	0.80	0.77	11394



Prediction without "Q61" feature

Model accuracy: 79.70%

	precision	recall	f1-score	support
1	0.67	0.35	0.46	2680
2	0.83	0.95	0.88	8714
accuracy			0.81	11394
macro avg	0.75	0.65	0.67	11394
weighted avg	0.79	0.81	0.78	11394



Prediction including "Q61" feature

Model accuracy: 80.60%

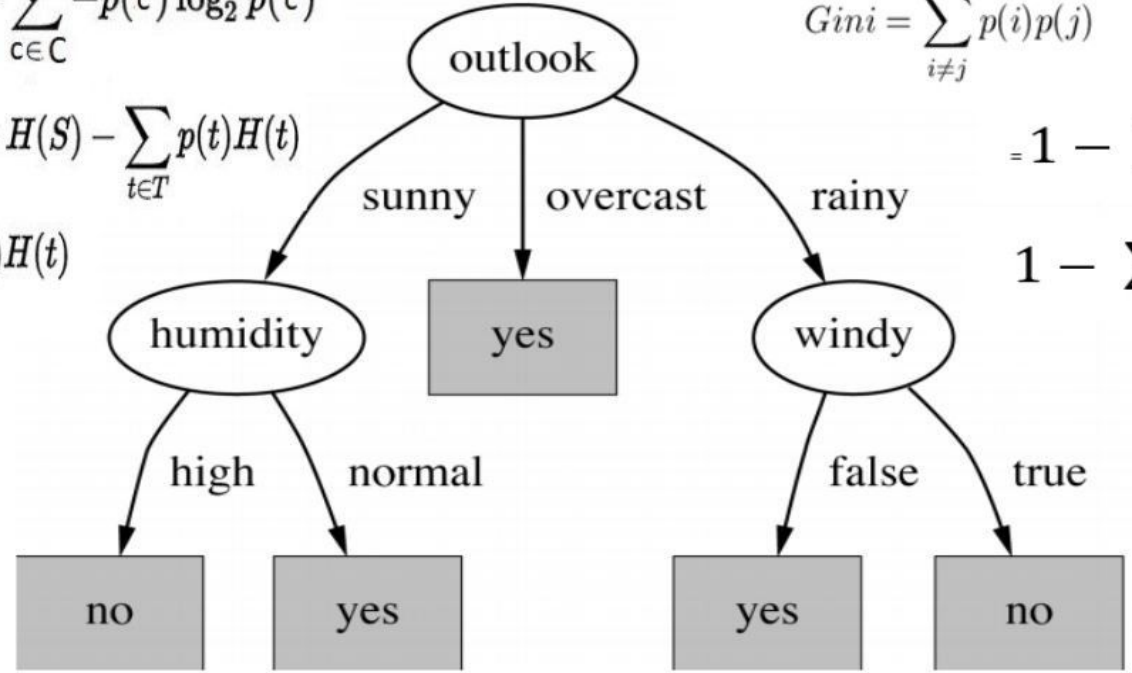
Decision Tree

Decision Tree | Math Formulation

$$H(S) = \sum_{c \in C} -p(c)\log_2 p(c)$$

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t)$$

$$\sum_{t \in T} p(t)H(t)$$



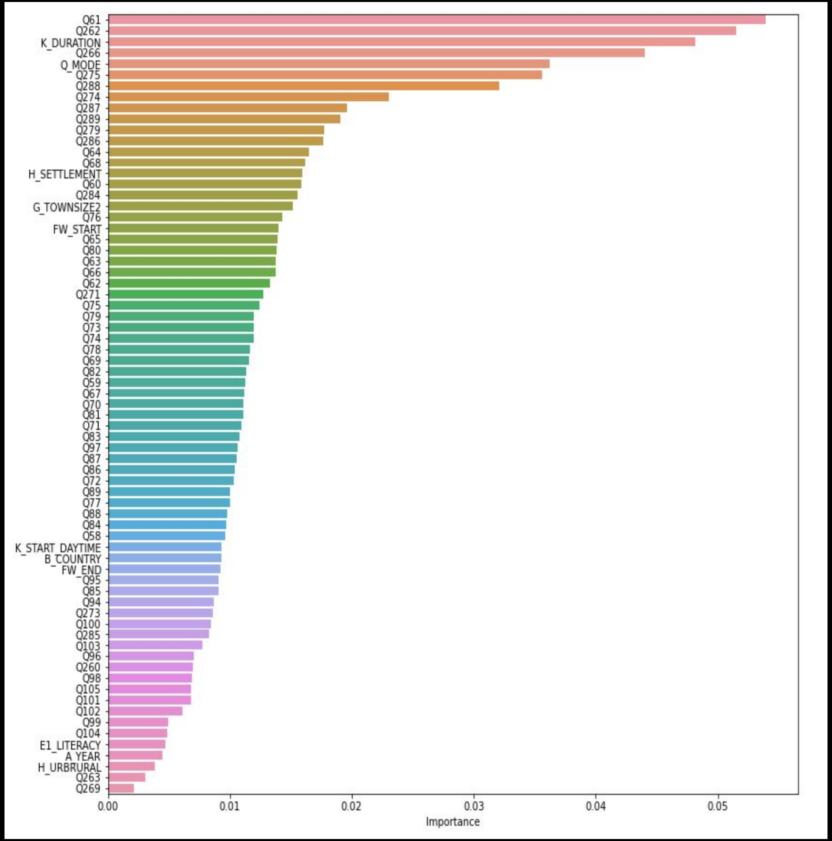
GINI Index

$$Gini = \sum_{i \neq j} p(i)p(j)$$

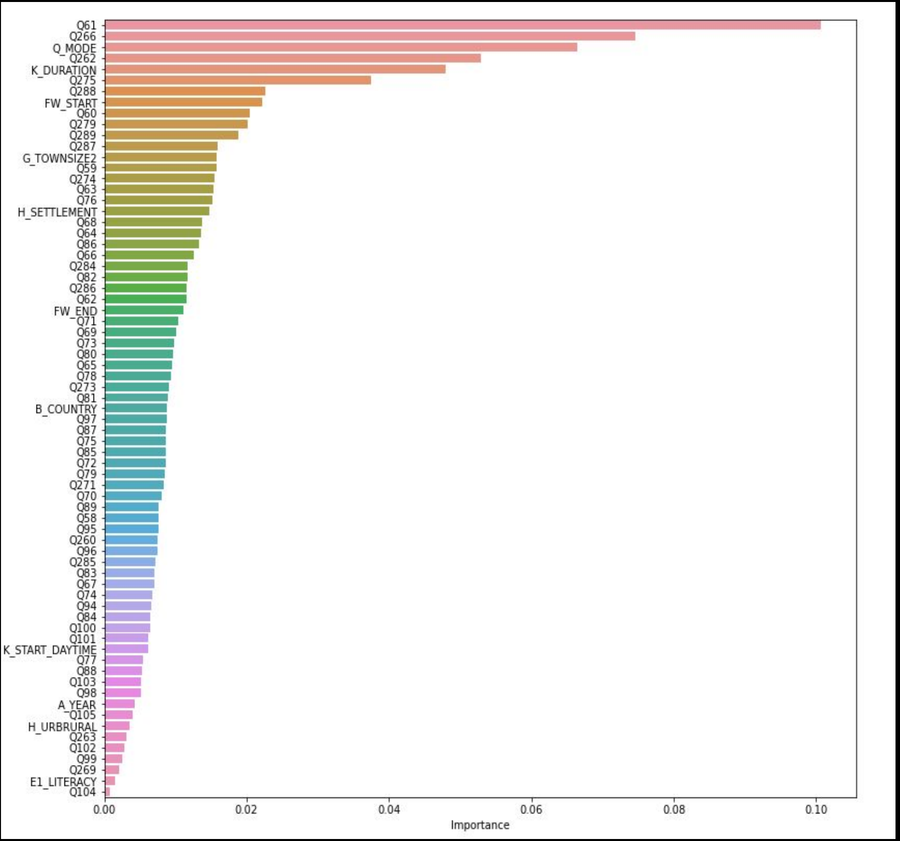
$$= 1 - \sum_{t=0}^{t=1} P_t^2$$

$$1 - \sum_{t=0}^{t=k} P_t^2$$

Decision Tree | Results



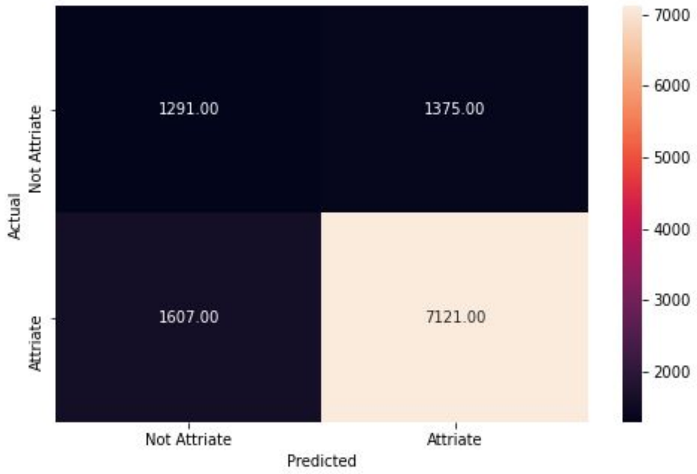
Before Grid Search



After Grid Search

Decision Tree | Confusion Matrix

	precision	recall	f1-score	support
1	0.45	0.48	0.46	2666
2	0.84	0.82	0.83	8728
accuracy			0.74	11394
macro avg	0.64	0.65	0.65	11394
weighted avg	0.75	0.74	0.74	11394

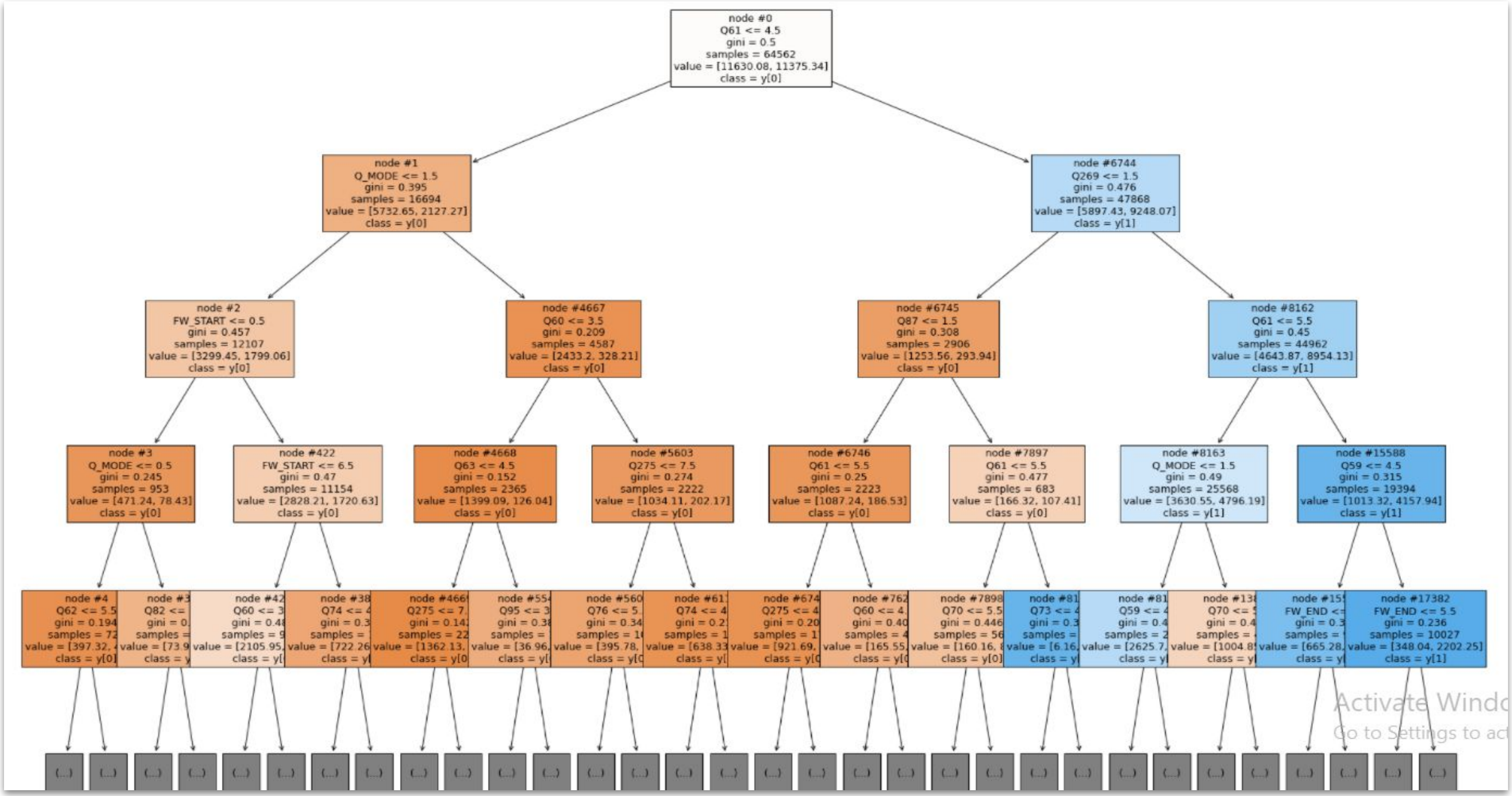


Before Grid Search

	precision	recall	f1-score	support
1	0.94	0.53	0.68	15104
2	0.87	0.99	0.93	49458
accuracy			0.88	64562
macro avg	0.91	0.76	0.80	64562
weighted avg	0.89	0.88	0.87	64562

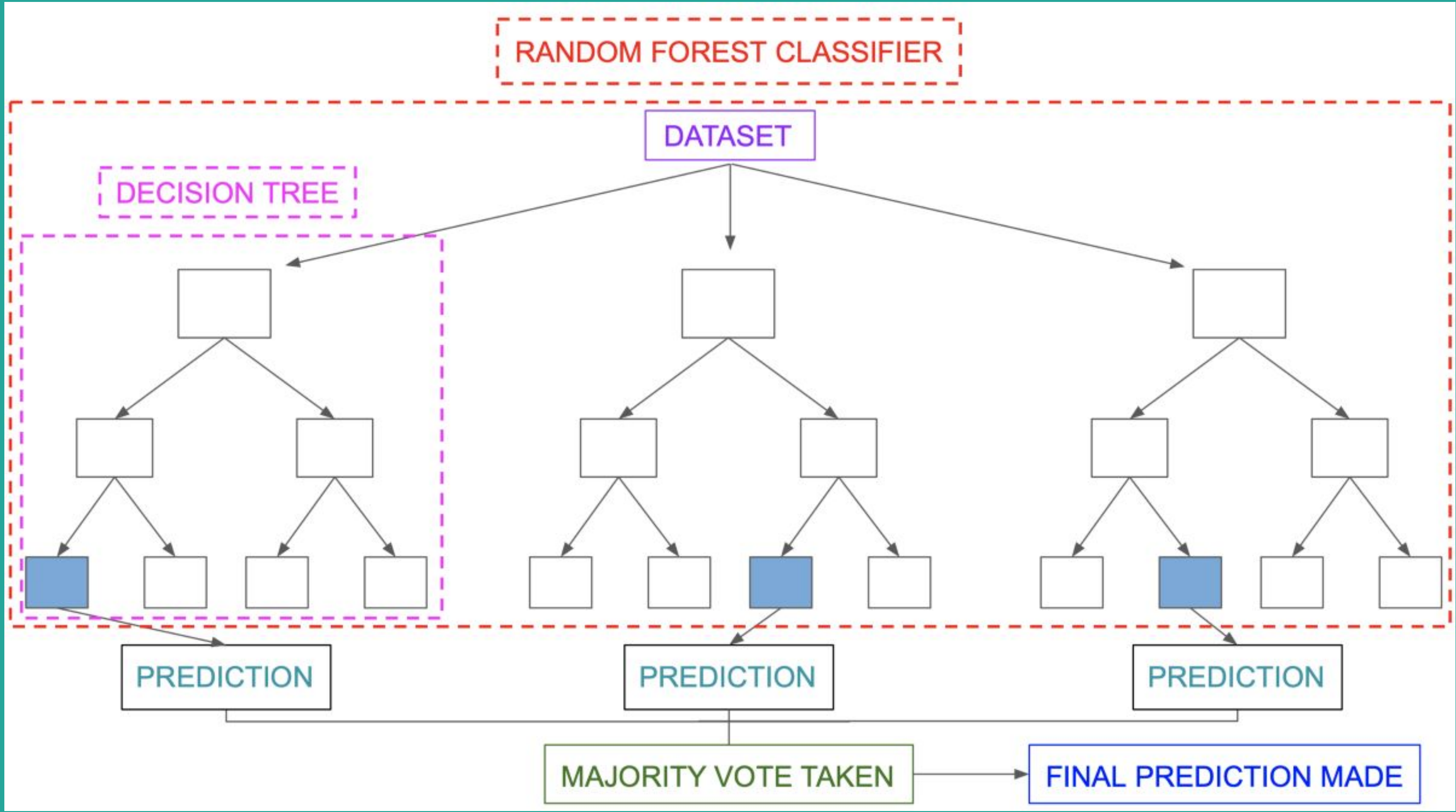
After Grid Search

Decision Tree

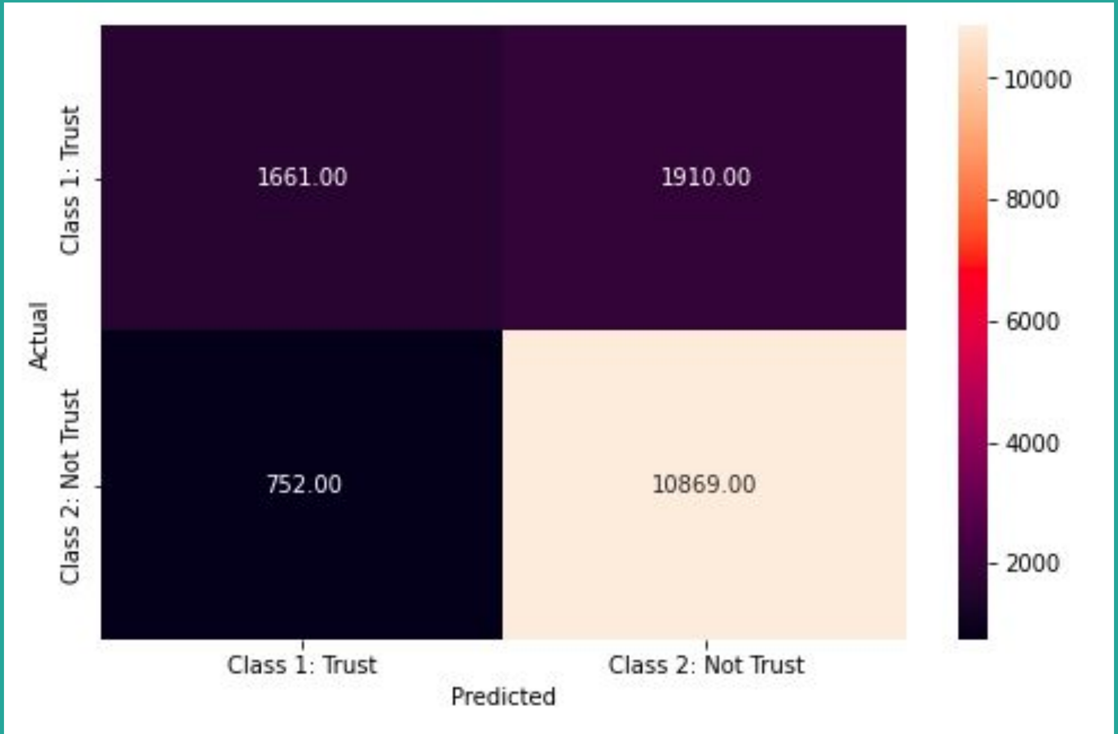


Random Forest

Random Forest



Random Forest | Results



Random Forest | Results

	precision	recall	f1-score	support
1	0.69	0.47	0.56	3571
2	0.85	0.94	0.89	11621
accuracy			0.82	15192
macro avg	0.77	0.70	0.72	15192
weighted avg	0.81	0.82	0.81	15192

Naive Bayes

Naive Bayes | Math formulation

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i \mid y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i \mid y)$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \dots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Categorical Naive Bayes

For each feature i in the training set X, categorical Naive Bayes estimates a categorical distribution for each feature i of X conditioned on the class y.
The index set of the samples is defined as $J = \{1, \dots, m\}$, with m as the number of samples.
The probability of category t in feature i given class c is estimated as:

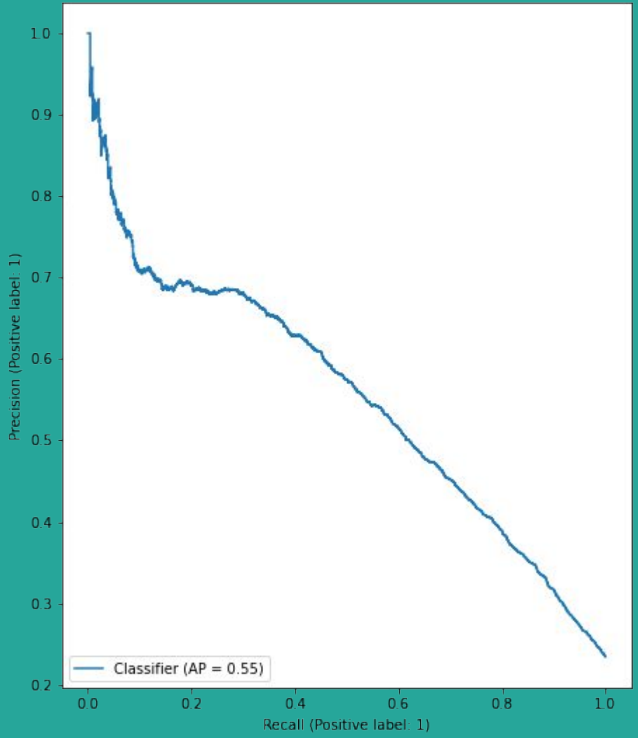
$$P(x_i = t \mid y = c; \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i}$$

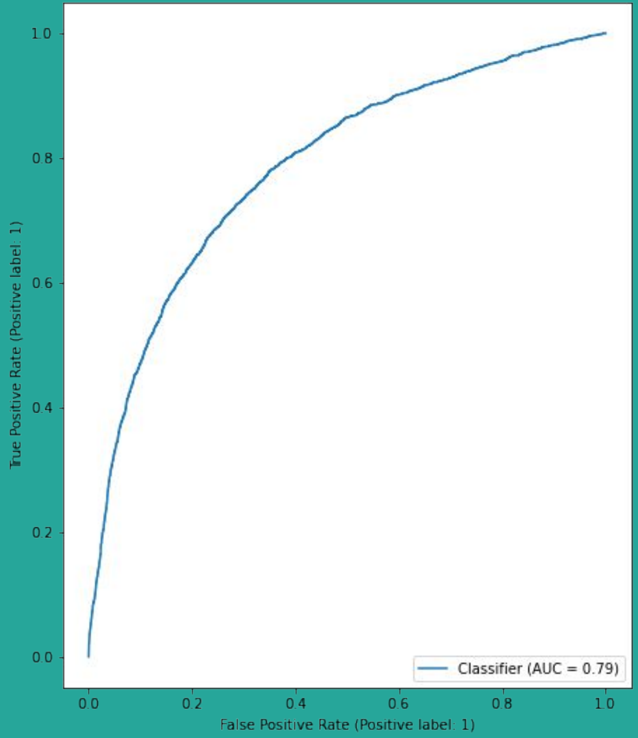
where $N_{tic} = |\{j \in J \mid x_{ij} = t, y_j = c\}|$ is the number of times category t appears in the samples x_i, which belong to class c,
$N_c = |\{j \in J \mid y_j = c\}|$ is the number of samples with class c, α is a smoothing parameter and n_i is the number of available categories of feature i.

Naive Bayes | Confusion Matrix

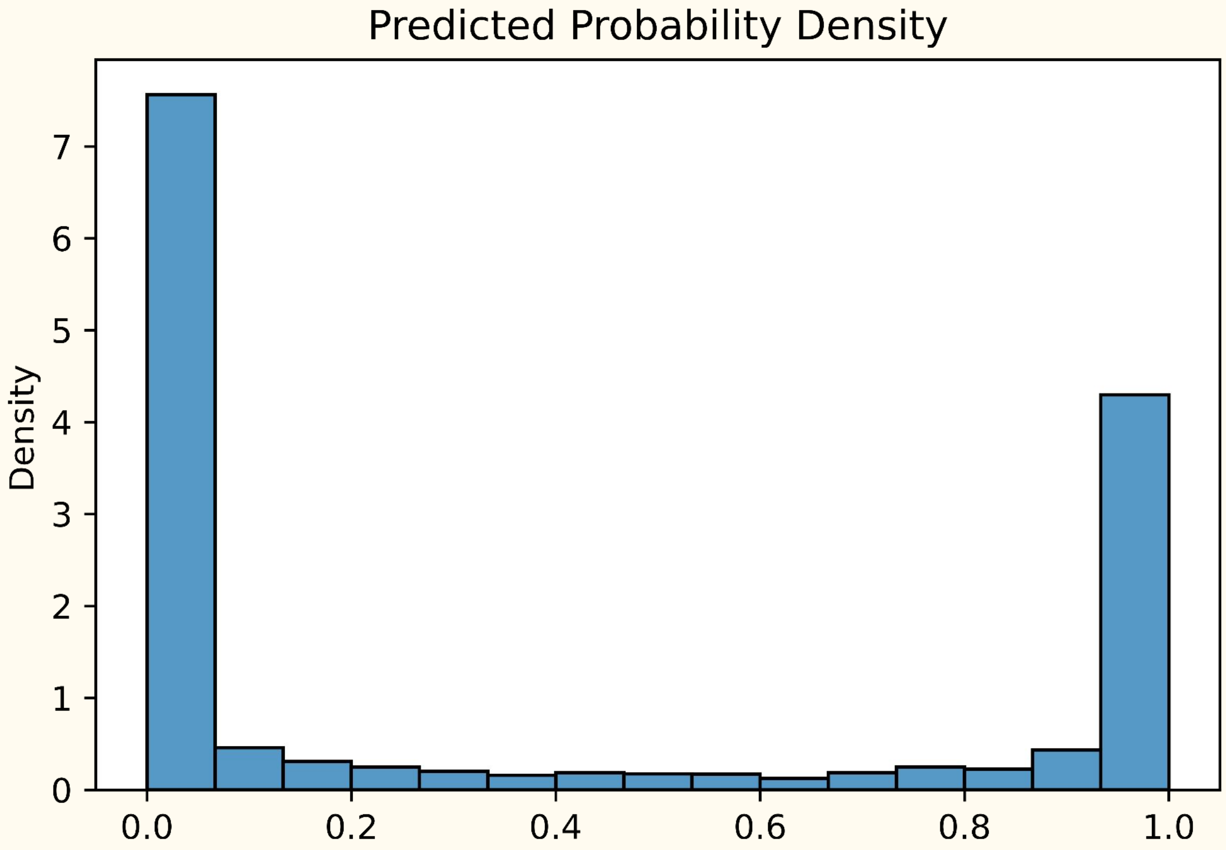
	precision	recall	f1-score	support
0	0.89	0.72	0.80	8728
1	0.44	0.72	0.54	2666
accuracy			0.72	11394
macro avg	0.67	0.72	0.67	11394
weighted avg	0.79	0.72	0.74	11394

Naive Bayes | Roc Curve





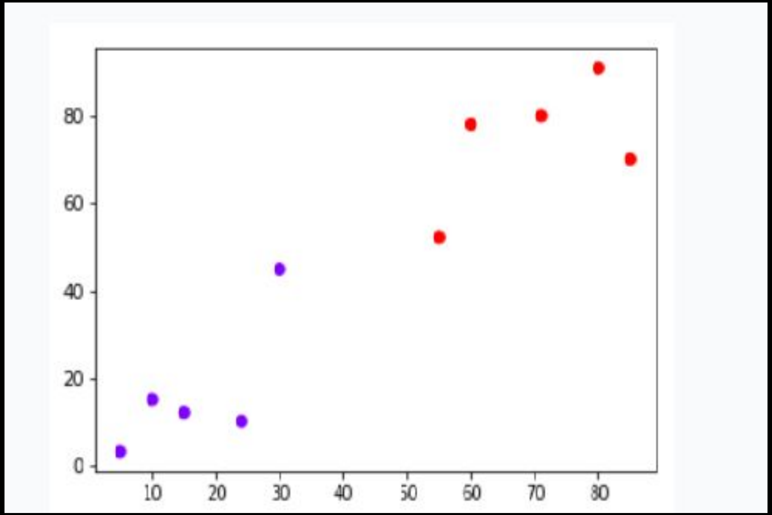
Naive Bayes | PMF

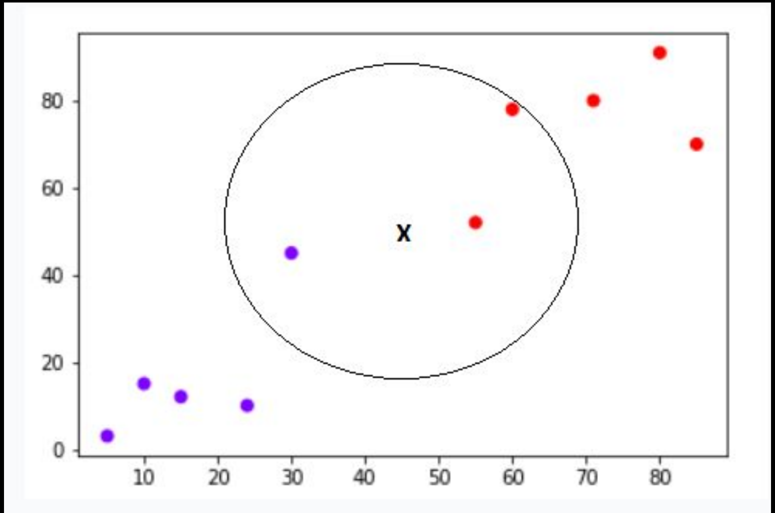


KNN

Found around the house!

The K-nearest neighbors (KNN) algorithm is a type of supervised machine learning algorithms.





Step 1: Calculate Euclidean Distance

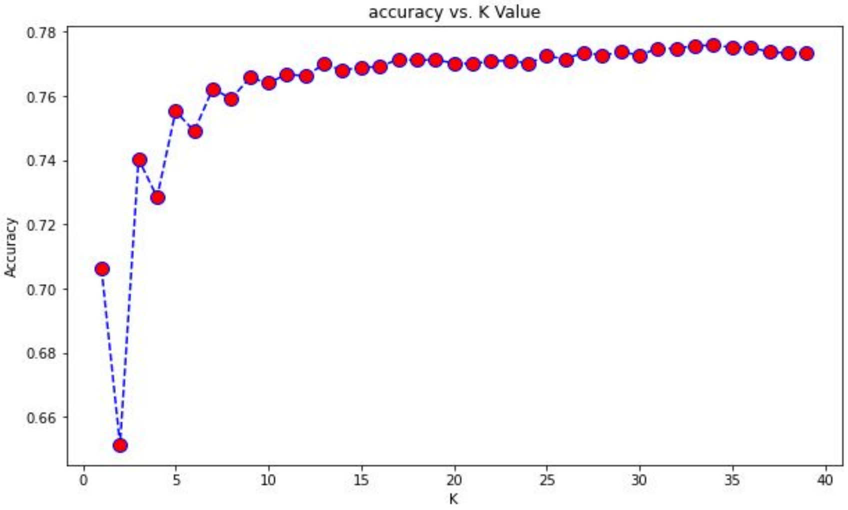
Euclidean Distance = sqrt(sum i to N (x1_i – x2_i)^2)

Step 2: Get Nearest Neighbors

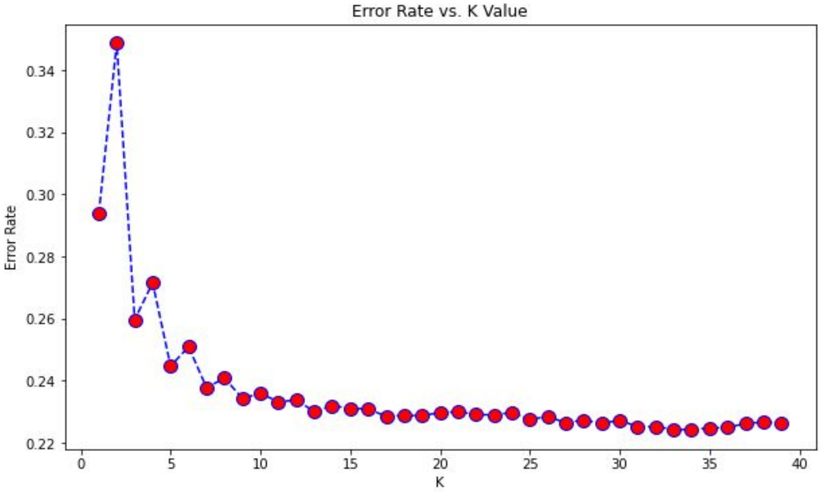
Step 3: Make Predictions

Optimal K value

Maximum accuracy:- 0.7757477243172952 at K = 33



Minimum error:- 0.2242522756827048 at K = 33



Knn | Results

	precision	recall	f1-score	support
1	0.58	0.31	0.40	3583
2	0.80	0.93	0.86	11605
accuracy			0.78	15380
macro avg	0.38	0.25	0.26	15380
weighted avg	0.74	0.78	0.75	15380

Models comparison

- **Logistic Regression**

	precision	recall	f1-score
1	0.67	0.35	0.46
2	0.83	0.95	0.88

- **Decision Tree**

	precision	recall	f1-score
1	0.94	0.53	0.68
2	0.87	0.99	0.93

- **Random Forest**

	precision	recall	f1-score
1	0.69	0.47	0.56
2	0.85	0.94	0.89

- **KNN**

	precision	recall	f1-score
1	0.58	0.31	0.40
2	0.80	0.93	0.86

- **Naive Bayes**

	precision	recall	f1-score
0	0.89	0.72	0.80
1	0.44	0.72	0.54

Metric	Formula
True positive rate, recall	$\frac{TP}{TP+FN}$
False positive rate	$\frac{FP}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
F-measure	$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Q&A