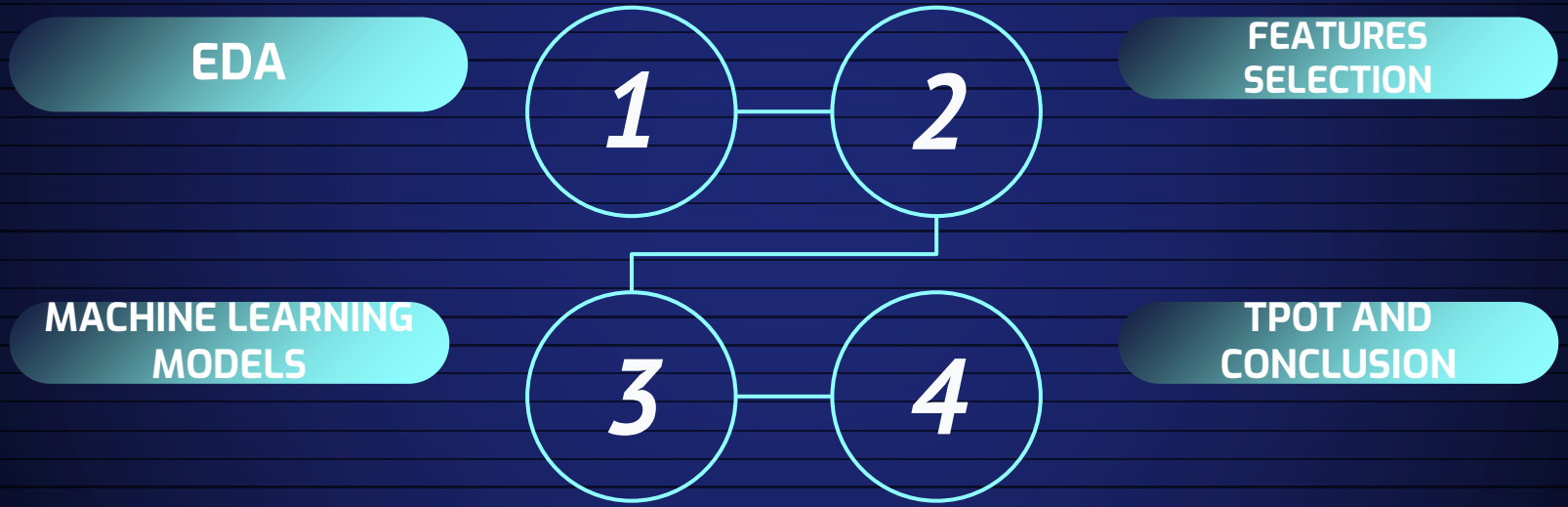
Project_7

MACHINE LEARNING SUPERVISED

AADIL BEN RACHID / BOUBACAR TRAORE


EDA
1
2
FEATURES SELECTION
MACHINE LEARNING MODELS
3
4
TPOT AND CONCLUSION

EDA

- 6819 rows
- 96 columns

- No missing values

- Numerical values

- A lot of collinearity between columns

- Unbalanced column Bankrupt?
  0 > 6599
  1 > 220

FEATURES SELECTION

```python
# Find the optimal number of features with SFM
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel

SFM = SelectFromModel(estimator=RandomForestClassifier())
s = SFM.fit(X, y)

n_features = s.transform(X).shape[1]
n_features
```

```
33
```

```python
# Get the features names
feature_idx = s.get_support()
feature_name = X.columns[feature_idx]
feature_name
```

```python
# split dataset with features
X = df[[' ROA(C) before interest and depreciation before interest',
       ' ROA(A) before interest and % after tax', ' Pre-tax net Interest Rate',
       ' Non-industry income and expenditure/revenue',
       ' Continuous interest rate (after tax)',
       ' Interest-bearing debt interest rate', ' Net Value Per Share (B)',
       ' Net Value Per Share (A)', ' Net Value Per Share (C)',
       ' Persistent EPS in the Last Four Seasons',
       ' Per Share Net profit before tax (Yuan ¥)', ' Net Value Growth Rate',
       ' Quick Ratio', ' Interest Expense Ratio',
       ' Total debt/Total net worth', ' Debt ratio %', ' Borrowing dependency',
       ' Net profit before tax/Paid-in capital', ' Average Collection Days',
       ' Fixed Assets Turnover Frequency', ' Working Capital to Total Assets',
       ' Cash/Total Assets', ' Cash/Current Liability',
       ' Inventory/Working Capital', ' Working Capital/Equity',
       ' Retained Earnings to Total Assets', ' Total income/Total expense',
       ' Net Income to Total Assets', ' Net Income to Stockholder\'s Equity',
       ' Liability to Equity', ' Degree of Financial Leverage (DFL)',
       ' Interest Coverage Ratio (Interest expense to EBIT)',
       ' Equity to Liability']]
y = df['Bankrupt?']

# train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)
```

MACHINE LEARNING MODELS

MACHINE LEARNING MODELS

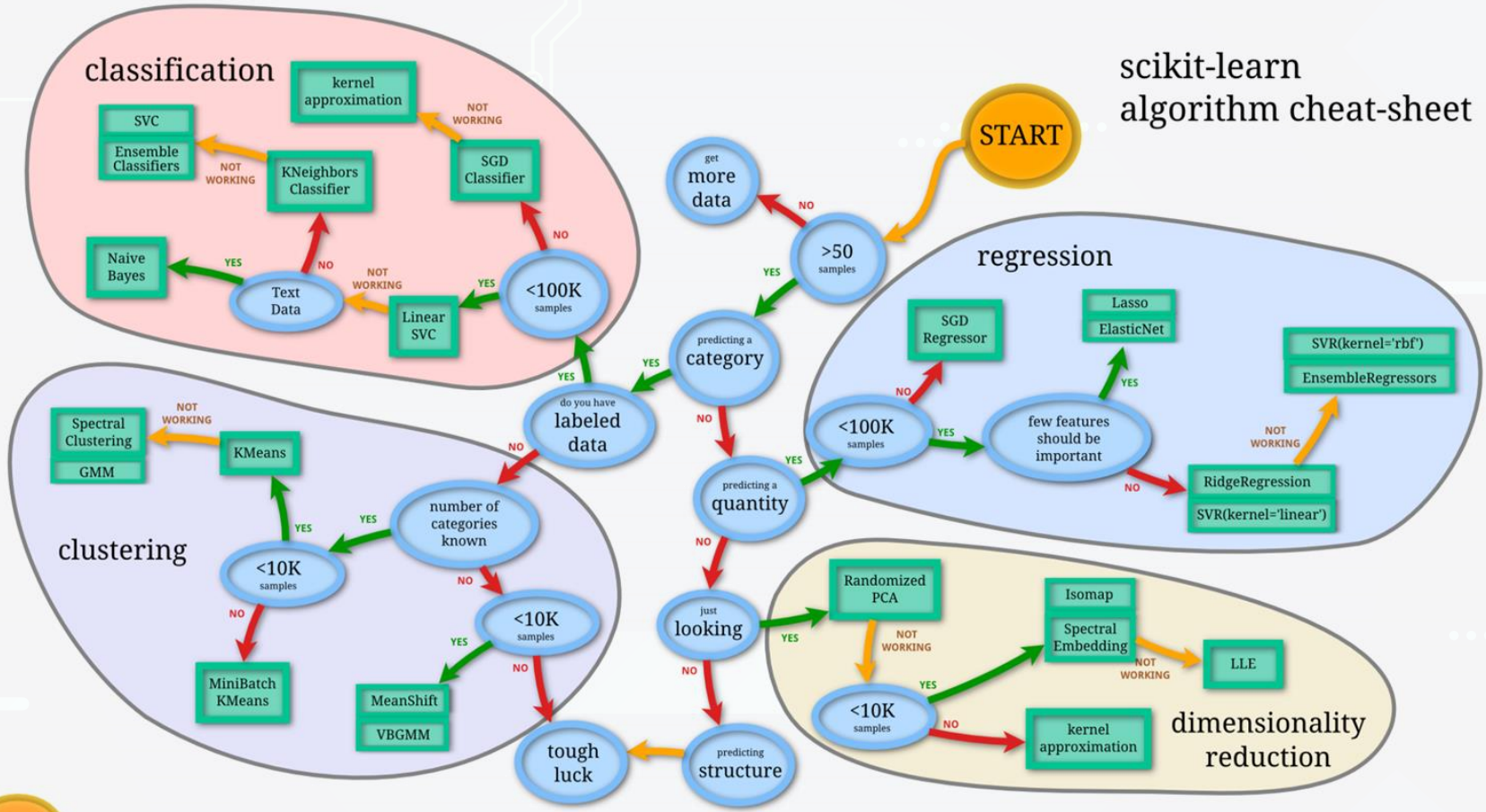
01 LINEAR SVC

02 KNN

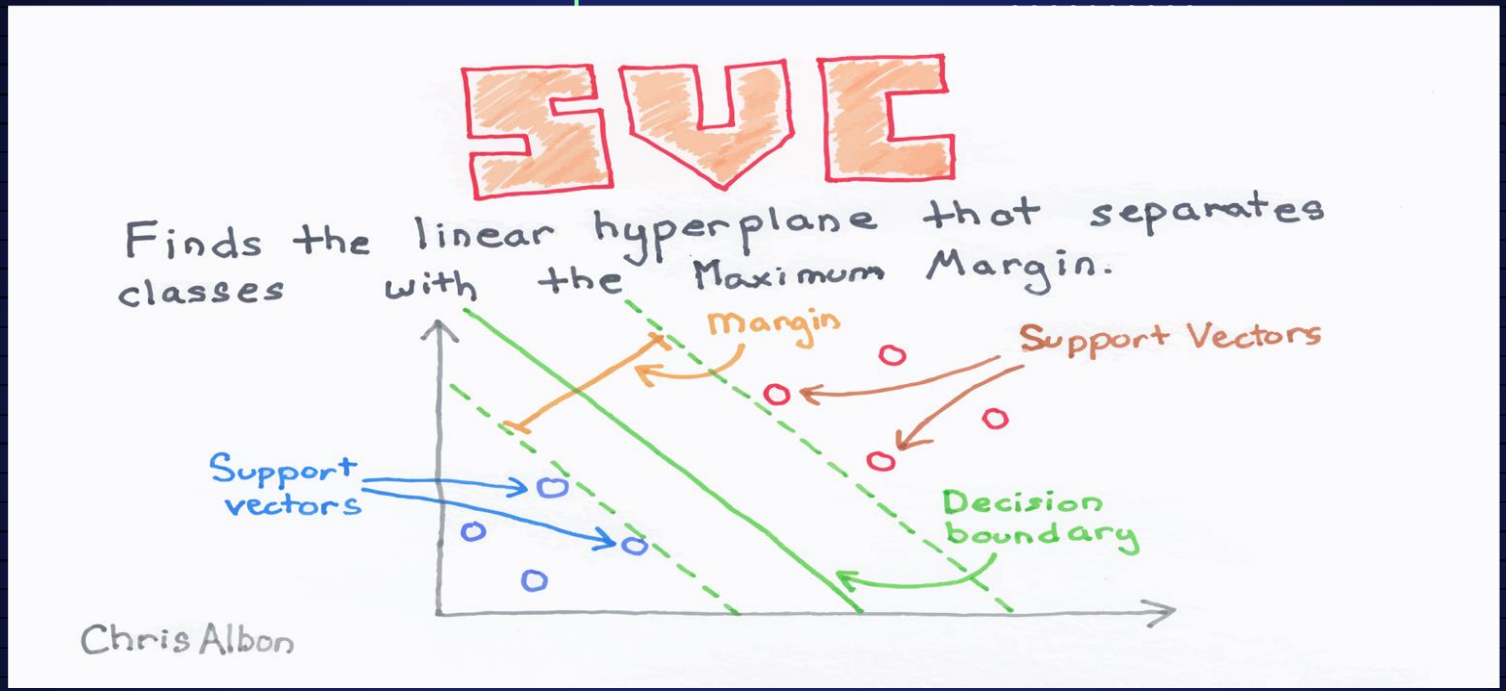
03 PASSIVE AGGRESSIVE CLASSIFIER

04 SGD CLASSIFIER

05 COMPLEMENT NB


scikit-learn
algorithm cheat-sheet
START
>50 samples
NO
get more data
YES
predicting a category
YES
NO
do you have labeled data
YES
<100K samples
YES
NOT WORKING
Linear SVC
NO
Text Data
YES
Naive Bayes
NO
KNeighbors Classifier
NOT WORKING
Ensemble Classifiers
SVC
NOT WORKING
SGD Classifier
NO
kernel approximation
classification
NO
predicting a quantity
YES
NO
<100K samples
YES
few features should be important
YES
NO
SGD Regressor
NO
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NOT WORKING
RidgeRegression
SVR(kernel='linear')
regression
NO
number of categories known
YES
<10K samples
YES
NOT WORKING
KMeans
Spectral Clustering
GMM
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
tough luck
clustering
NO
just looking
YES
Randomized PCA
YES
NOT WORKING
<10K samples
YES
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
dimensionality reduction
NO
predicting structure
Back
scikit learn

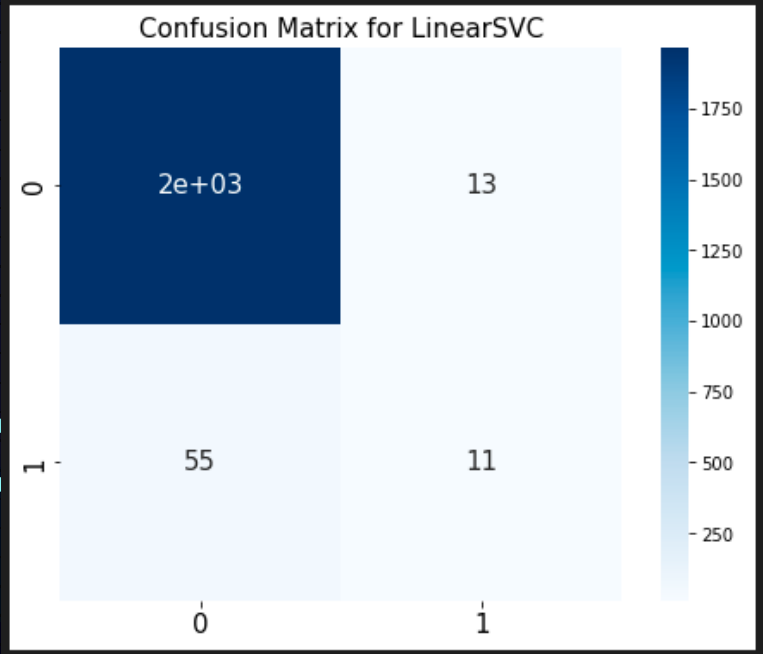
LINEAR SVC



class **sklearn.svm.LinearSVC**(*penalty='l2', loss='squared_hinge', *, dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000*)

The classification report for LinearSVC is:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	1980
1	0.46	0.17	0.24	66
accuracy			0.97	2046
macro avg	0.72	0.58	0.61	2046
weighted avg	0.96	0.97	0.96	2046



Good accuracy 97%

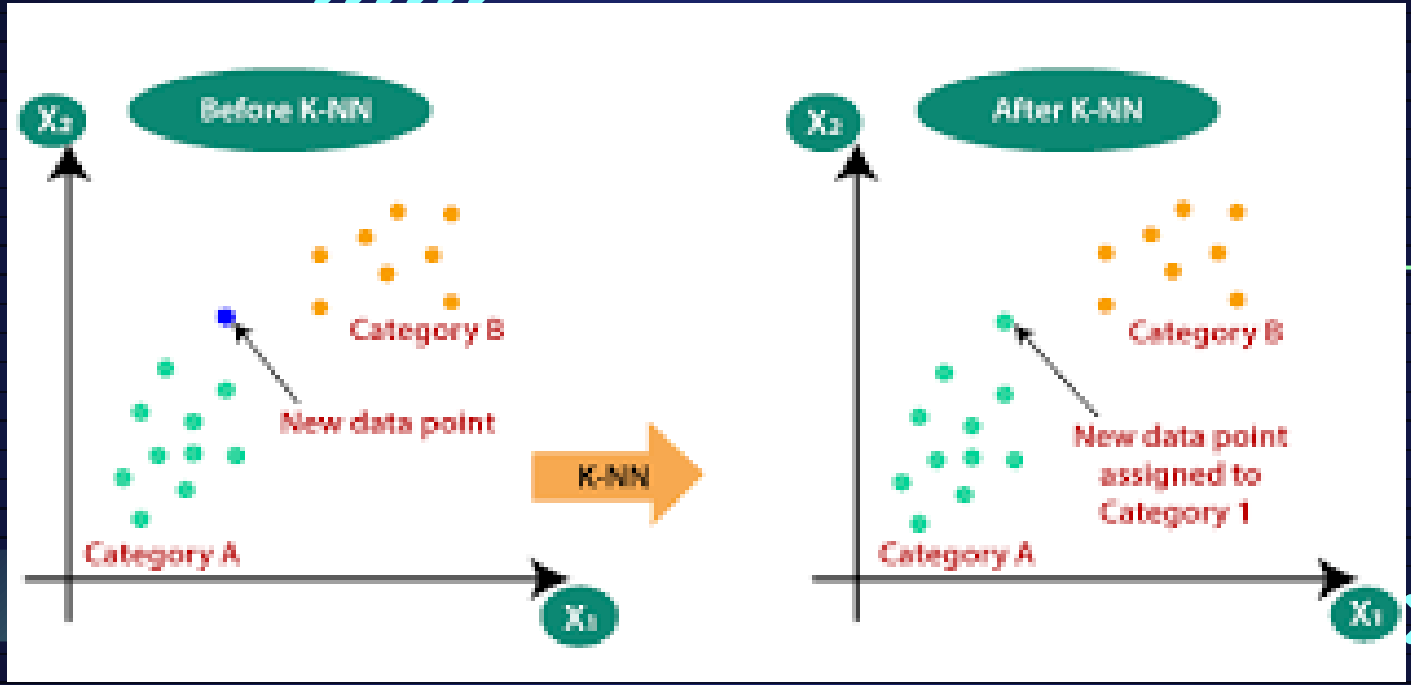
But

Bad precision 47%

And

Very low recall 17%

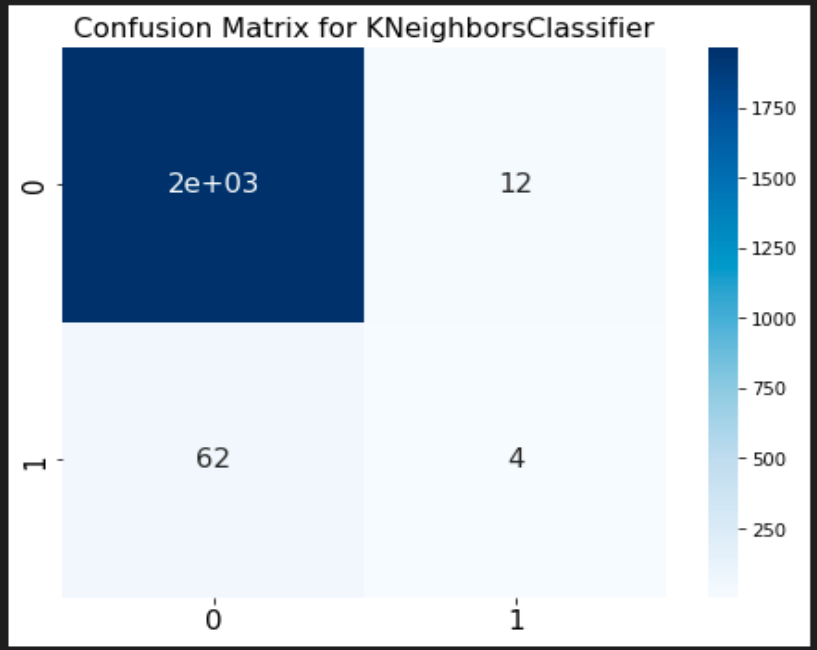
K NEAREST NEIGHBOR CLASSIFIER



class **sklearn.neighbors.KNeighborsClassifier**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*) [source]

The classification report for KNeighborsClassifier is:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	1980
1	0.25	0.06	0.10	66
accuracy			0.96	2046
macro avg	0.61	0.53	0.54	2046
weighted avg	0.95	0.96	0.95	2046



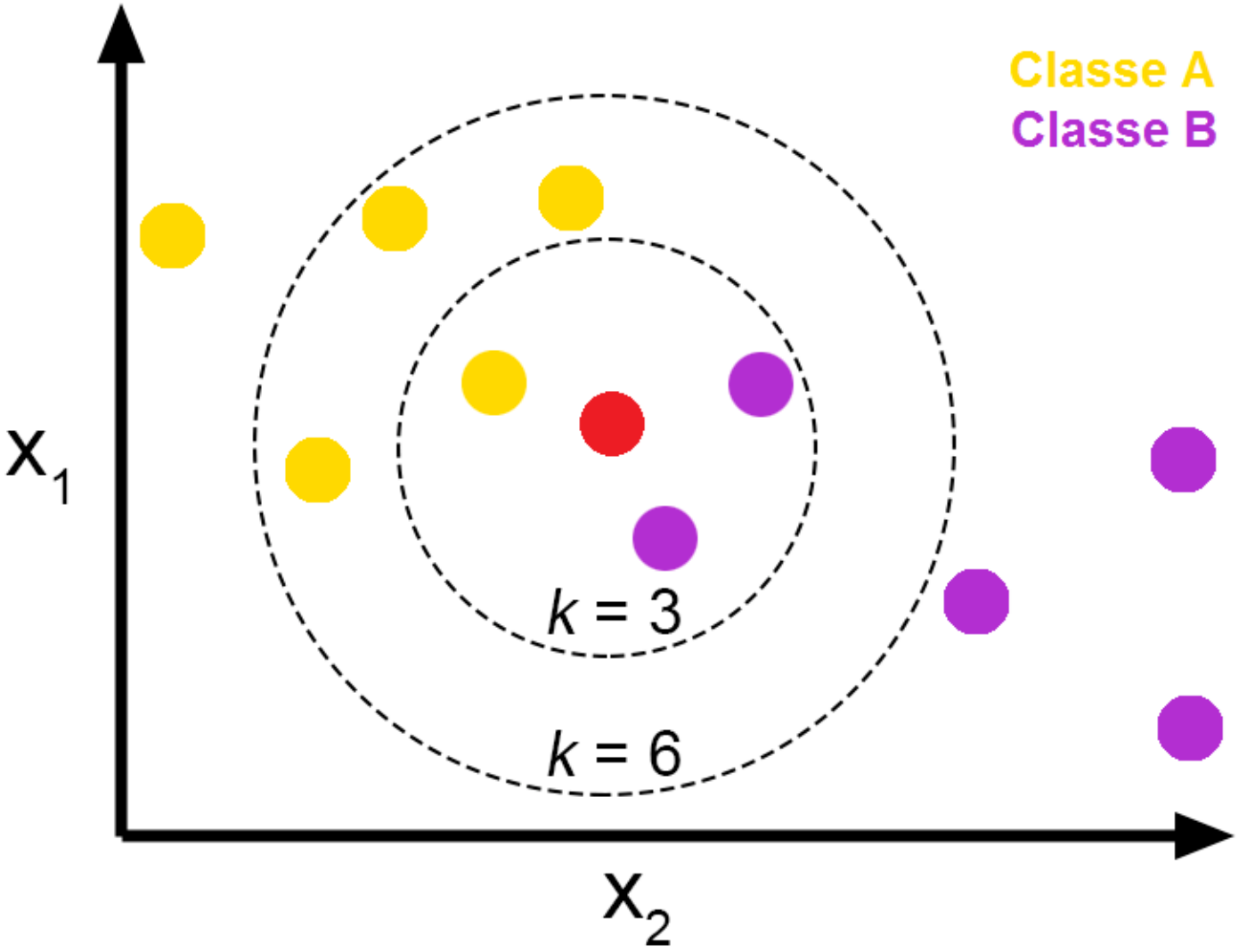
Good accuracy 96%

But

Bad precision 25%

And

Very low recall 6%

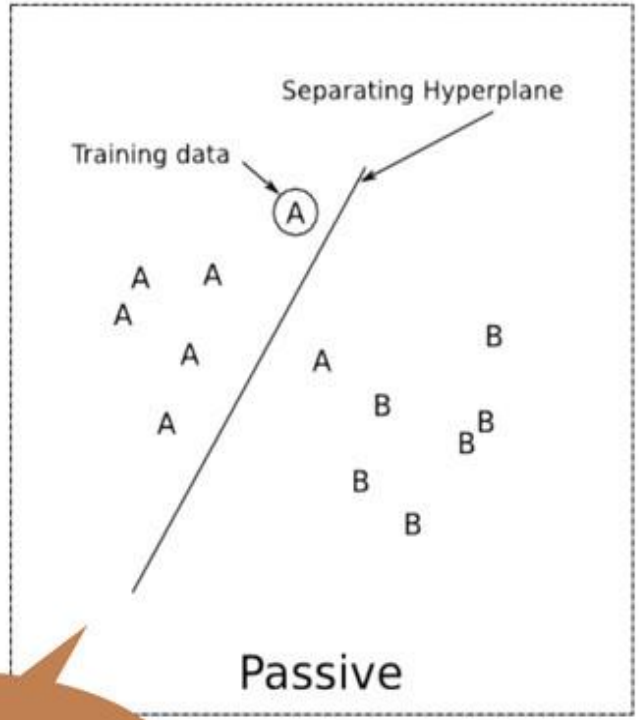

Classe A
Classe B
x_1
x_2
$k = 3$
$k = 6$

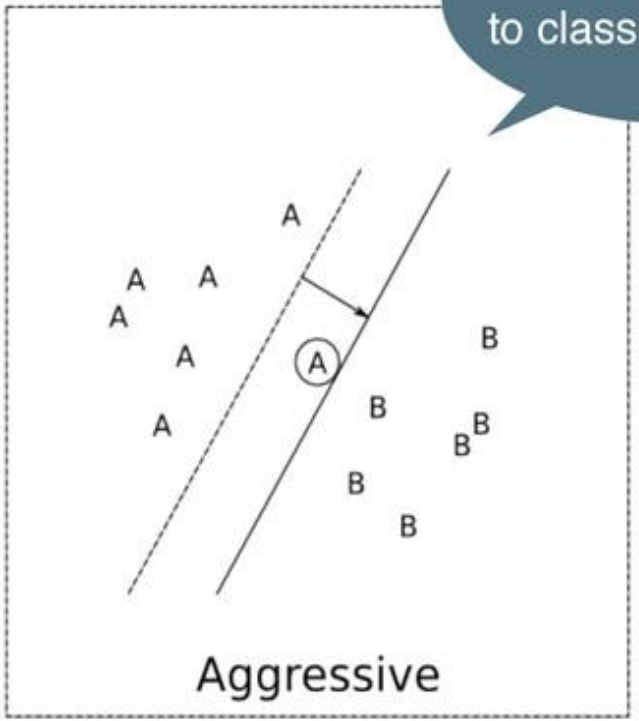
PASSIVE AGGRESSIVE CLASSIFIER

Passive-Aggressive algorithms are called so because :
Passive: If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model.
Aggressive: If the prediction is incorrect, make changes to the model.

class **sklearn.linear_model.PassiveAggressiveClassifier**(*, *C=1.0, fit_intercept=True, max_iter=1000, tol=0.001, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, shuffle=True, verbose=0, loss='hinge', n_jobs=None, random_state=None, warm_start=False, class_weight=None, average=False*)

Passive & Aggressive: Illustrated



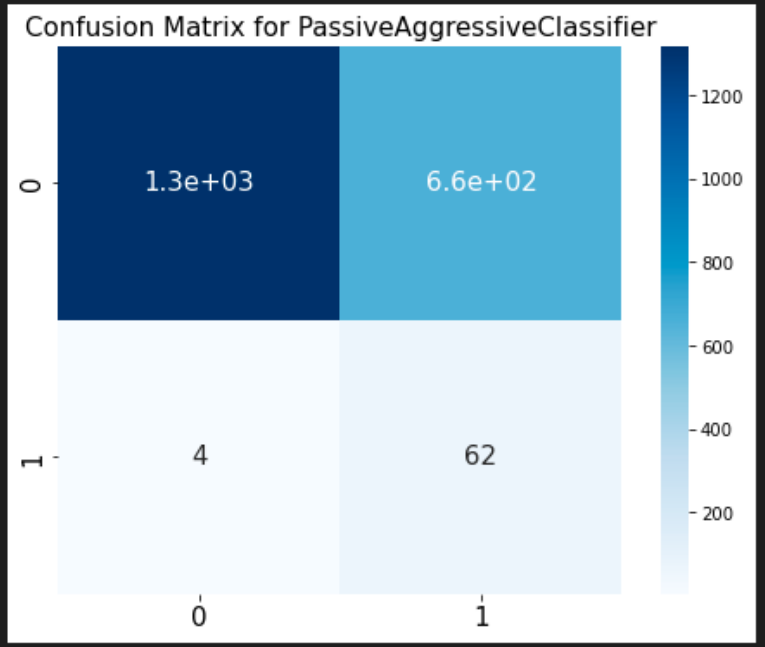


Move **minimally** to classify correctly.

Do nothing.

The classification report for PassiveAggressiveClassifier is:

	precision	recall	f1-score	support
0	1.00	0.67	0.80	1980
1	0.09	0.94	0.16	66
accuracy			0.67	2046
macro avg	0.54	0.80	0.48	2046
weighted avg	0.97	0.67	0.78	2046



Good accuracy 67%

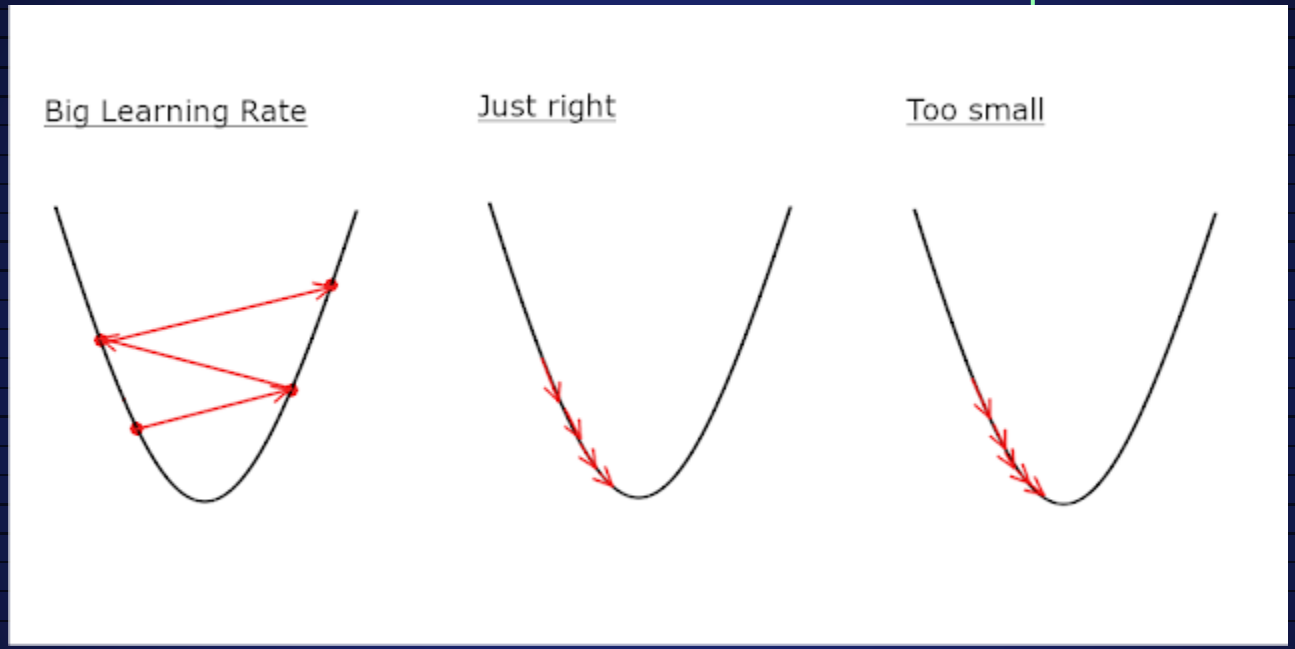
But

Very bad precision 9%

And

Very good recall 94%

SGD CLASSIFIER



class **sklearn.linear_model.SGDClassifier**(*loss='hinge', *, penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, n_jobs=None, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False, average=False*) [source]

The advantages of Stochastic Gradient Descent are:

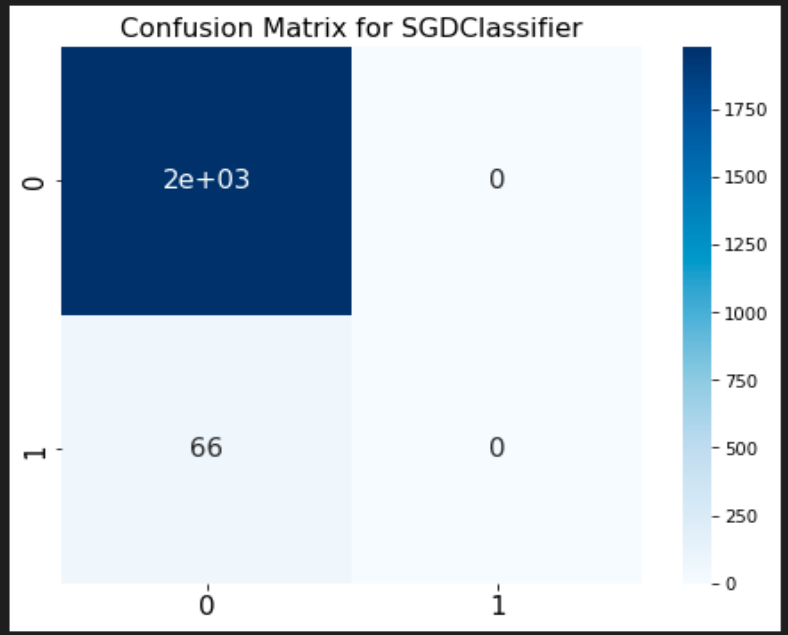
- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- SGD requires a number of hyperparameters including the regularization parameter and the number of iterations.
- SGD is sensitive to feature scaling.

The classification report for SGDClassifier is:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	1980
1	0.00	0.00	0.00	66
accuracy			0.97	2046
macro avg	0.48	0.50	0.49	2046
weighted avg	0.94	0.97	0.95	2046



Good accuracy 97%

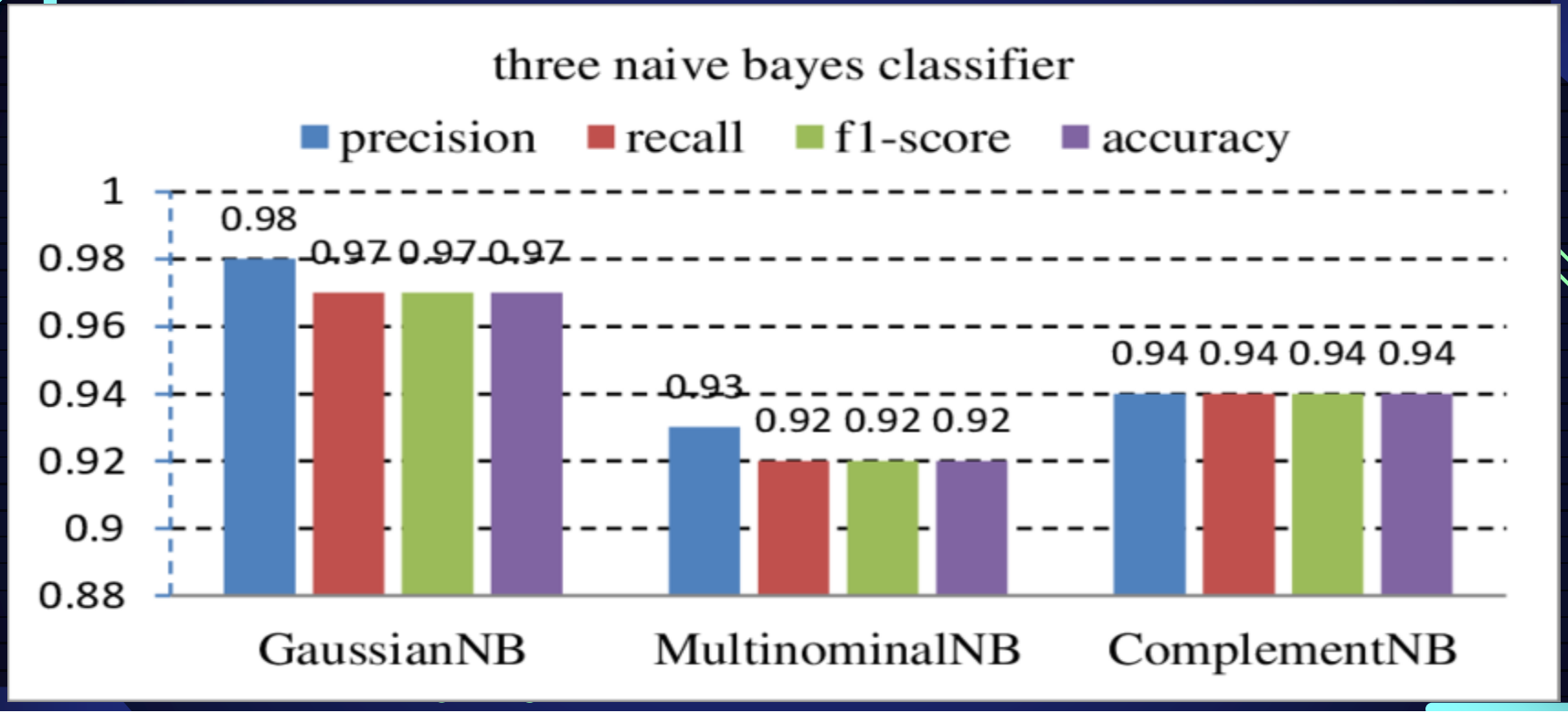
But

Very bad precision 0%

And

Very bad recall 0%

COMPLEMENT NB



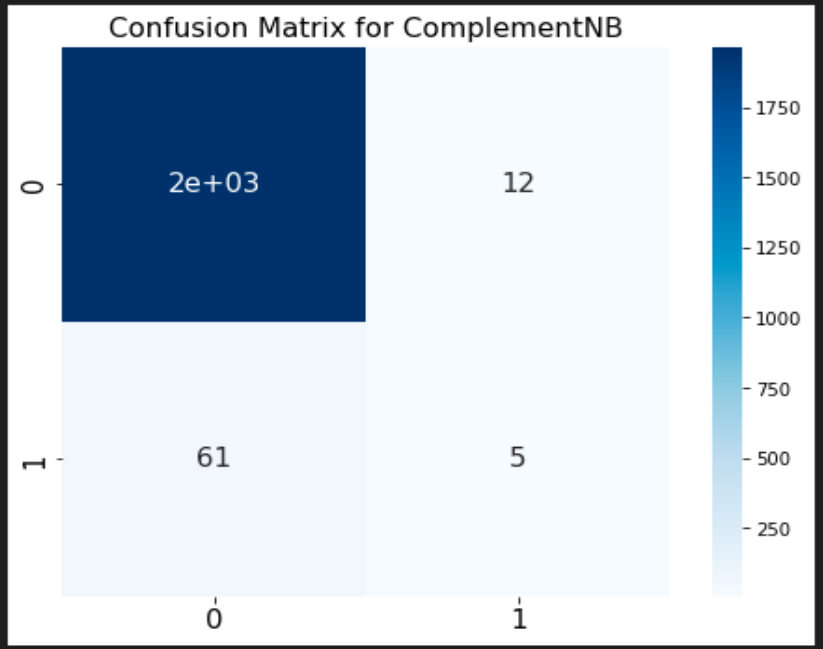
class `sklearn.naive_bayes.ComplementNB`(*, *alpha=1.0, fit_prior=True, class_prior=None, norm=False*)

The Complement Naive Bayes classifier was designed to **correct the "severe assumptions" made by the standard Multinomial Naive Bayes classifier**. It is particularly suited for imbalanced data sets

Multinomial Naive Bayes algorithm is **a probabilistic learning method that is mostly used in Natural Language Processing (NLP)**. The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article

The classification report for ComplementNB is:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	1980
1	0.29	0.08	0.12	66
accuracy			0.96	2046
macro avg	0.63	0.53	0.55	2046
weighted avg	0.95	0.96	0.95	2046



Good accuracy 97%

But

Very low precision 29%

And

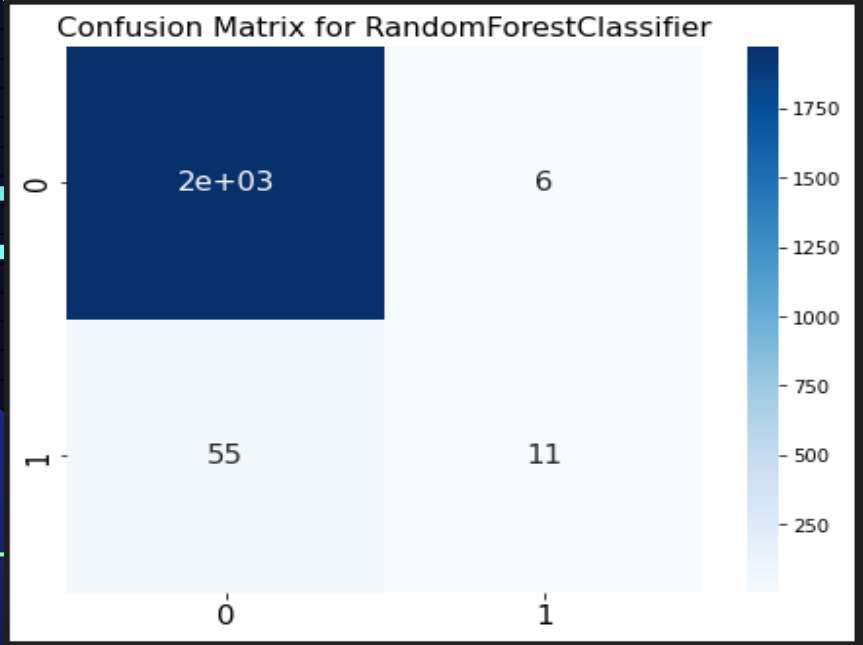
Bad recall 8%

TPOT RESULT

Random Forest Classifier

The classification report for RandomForestClassifier is:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	1980
1	0.65	0.17	0.27	66
accuracy			0.97	2046
macro avg	0.81	0.58	0.62	2046
weighted avg	0.96	0.97	0.96	2046



Good accuracy 97%

Precision 65%

Very low recall 17%

MODELS RESULTS RECAP

	Knn	LinearSVC	Passive Agressive	SGD	ComplementNB	Random Forest
accuracy	96	97	67	97	97	97
precision	25	45	9	0	29	65
recall	6	17	94	0	8	17

Challenges
Improvements

Features selection

Use SMOTEN for oversampling

Understanding the models

Cross validation

Hyper parameter tuning in parallel

Make slides