DOCUMENT CLASSIFICATION MODELING

INTRODUCTION & OVERVIEW

Gaussian Naive Bayes model is a machine learning classification model that uses normally distributed features and probabilities to predict the output variable or class. This program classifies documents into 4 types: Invoice, Packing List, Bill of Lading, and Other. It uses the following steps to classify documents:

1. Nanonets Optical Character Recognition Package to identify words in document
2. Creation of input CSV file for the classification model
3. Identification of best model through pyCaret library
4. Training and Testing of the Gaussian Naive Bayes model
5. Output and Prediction

In order to create the Naive Bayes model, pyCaret (a low code machine learning library) was used to compare and identify the different classification models that could be used. After running many data tables though pyCaret, it identified that Gaussian Naive Bayes was the best binary classification model to use due to the nature and characteristics of the table.

Note: pip install ocr-nanonets-wrapper

You can use pip to install the nanonets OCR package (python 3 is required for this)

DATA COLLECTION AND CLEANING

Data Collection Through PDFs

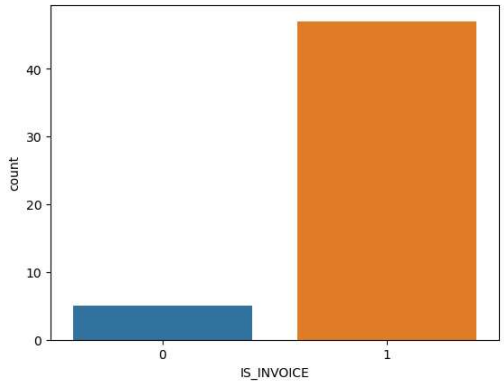
The initial data table (CSV) called NBC-1.csv was created manually by going through the given known PDFs: 1. Invoice (Scanned).pdf : 11 pages of Invoices 2. mcc-34page.pdf : 34 pages of Invoices 3. inv + pkg + bl 1.pdf : 1 Invoice, 1 Packing List, 1 Bill of Lading, 1 other 4. inv + pkg + bl 2.pdf : 1 Invoice, 1 Bill of Lading, 1 Packing List

The NBC-1.csv table contains only binary data (True/False or 1/0). Data regarding the presence of the following were collected and written to the table: Serial number, name of PDF, page number, invoice heading, packing list heading, bill of lading heading, details of container, number of packages, description, quantity, price, net amount/price, gross weight, net weight, and measurement details. Finally, the pages of the known PDFs were marked as Invoice, Packing List, or Bill of Lading.

Documents can be classified into Invoice, Packing List, or Bill of Lading depending on the presence of the previously mentioned details in the PDFs/ images. An overview of the CSV file (NBC-1.csv) is given further below.

This table was cleaned by removing any anomalies was made into the proper format using both pandas library and Excel (due to the small amount of available data). The initial table (training_data_csv-Copy1.csv) consisted of only data from the given PDFs and was unbalanced (leading to the recommendation of a Gradient Boosted Classifier Model for the limited data).

In [11]:
```
# Visual representation of unbalanced data biased towards Invoice classification
import seaborn as sns
df = pd.read_csv("training_data_csv-Copy1.csv")
ax = sns.countplot(x = "IS_INVOICE", data = df)
```



Since the table was unbalanced and biased towards invoice predictions, more packing list and bill of lading data values were added to even out and balance the training data for the machine learning model. Hence, the final data table for training the model was created: NBC-1.csv. Here is an overview of the NBC-1.csv:

In [13]:
```
# Display of NBC-1.csv table that is used to train models for prediction
import pandas as pd
data = pd.read_csv("NBC-1.csv")
data
```

Out[13]:

	SRL_NO	PDF_NAME	PAGE_NUMBER	IL_HEADING	PL_HEADING	BOL_HEADING	CONTAINER_DETAILS	NO_OF_PCKGS	DESCRIPTION	QTY	PRICE	NET_AMOUNT	GROSS_WEIGHT	NET_WEIGHT	MEASUREMENT	IS_INVOICE	IS_PL	IS_BOL
0	1	Invoice (Scanned)	1	True	True	False	True	True	True	True	True	True	False	False	False	1	0	0
1	2	Invoice (Scanned)	2	True	True	False	True	True	True	True	True	True	False	False	False	1	0	0
2	3	Invoice (Scanned)	3	True	True	False	True	True	True	True	True	True	False	False	False	1	0	0
3	4	Invoice (Scanned)	4	True	True	False	True	True	True	True	True	True	False	False	False	1	0	0
4	5	Invoice (Scanned)	5	True	True	False	True	True	True	True	True	True	False	False	False	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
65	66	BOL extended	4	False	False	True	True	True	True	False	False	False	True	False	True	0	0	1
66	67	BOL extended	5	False	False	True	True	True	True	False	False	False	True	False	True	0	0	1
67	68	BOL extended	6	False	False	True	True	True	True	False	False	False	True	False	True	0	0	1

IDENTIFYING THE BEST ML CLASSIFICATION MODEL: GAUSSIAN NAIVE BAYES

pyCaret Model Identification

In [16]:
```
# pyCaret classification setup
import pycaret
from pycaret.classification import *
s = setup(data, target = 'IS_INVOICE', session_id = 123)
```

	Description	Value
0	Session id	123
1	Target	IS_INVOICE
2	Target type	Binary
3	Original data shape	(70, 18)
4	Transformed data shape	(70, 23)
5	Transformed train set shape	(49, 23)
6	Transformed test set shape	(21, 23)
7	Numeric features	4
8	Categorical features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	f152

In [17]:
```
from pycaret.classification import ClassificationExperiment
exp = ClassificationExperiment()
# init setup (on exp)
exp.setup(data, target = 'IS_INVOICE', session_id = 123)
```

	Description	Value
0	Session id	123
1	Target	IS_INVOICE
2	Target type	Binary
3	Original data shape	(70, 18)
4	Transformed data shape	(70, 23)
5	Transformed train set shape	(49, 23)
6	Transformed test set shape	(21, 23)
7	Numeric features	4
8	Categorical features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	5c9f

Out[17]:
```
<pycaret.classification.oop.ClassificationExperiment at 0x226b8e4b8d0>
```

In [18]:
```
# Comparing the baseline models uwing features like Precision, Recall, F1 score, and runtime
best = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0220
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0200
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0740
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0180
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0340
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0570
catboost	CatBoost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9050
knn	K Neighbors Classifier	0.9800	0.9875	0.9750	1.0000	0.9857	0.9545	0.9612	0.0610
ridge	Ridge Classifier	0.9800	1.0000	0.9750	1.0000	0.9857	0.9545	0.9612	0.0220
lda	Linear Discriminant Analysis	0.9800	1.0000	1.0000	0.9750	0.9857	0.9545	0.9612	0.0180
lr	Logistic Regression	0.9600	1.0000	0.9750	0.9750	0.9714	0.9091	0.9225	0.8300
lightgbm	Light Gradient Boosting Machine	0.9600	1.0000	0.9750	0.9750	0.9714	0.9091	0.9225	0.0520
svm	SVM - Linear Kernel	0.9400	1.0000	0.9083	1.0000	0.9457	0.8776	0.8946	0.0210
dummy	Dummy Classifier	0.6750	0.5000	1.0000	0.6750	0.8024	0.0000	0.0000	0.0260
qda	Quadratic Discriminant Analysis	0.4250	0.0000	0.5000	0.3000	0.3750	0.0000	0.0000	0.0210

In [19]:
```
# compare models using OOP
exp.compare_models()
```

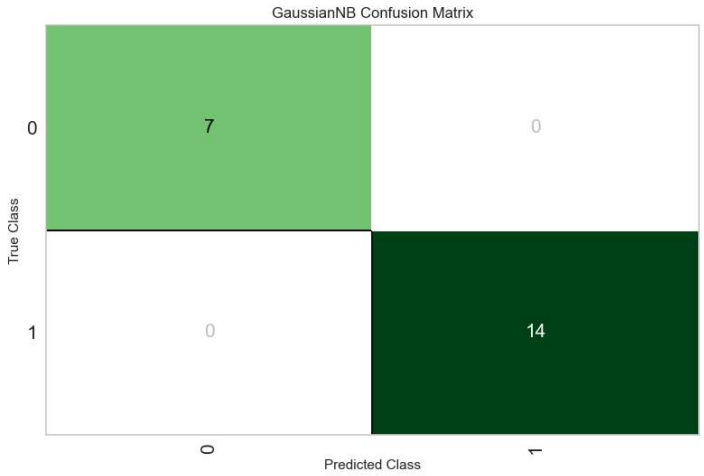
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
nb	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0210
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0210
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0650
ada	Ada Boost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0200
gbc	Gradient Boosting Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0370
et	Extra Trees Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0560
catboost	CatBoost Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.7660
knn	K Neighbors Classifier	0.9800	0.9875	0.9750	1.0000	0.9857	0.9545	0.9612	0.0620
ridge	Ridge Classifier	0.9800	1.0000	0.9750	1.0000	0.9857	0.9545	0.9612	0.0190
lda	Linear Discriminant Analysis	0.9800	1.0000	1.0000	0.9750	0.9857	0.9545	0.9612	0.0240
lr	Logistic Regression	0.9600	1.0000	0.9750	0.9750	0.9714	0.9091	0.9225	0.0210
lightgbm	Light Gradient Boosting Machine	0.9600	1.0000	0.9750	0.9750	0.9714	0.9091	0.9225	0.0480
svm	SVM - Linear Kernel	0.9400	1.0000	0.9083	1.0000	0.9457	0.8776	0.8946	0.0180
dummy	Dummy Classifier	0.6750	0.5000	1.0000	0.6750	0.8024	0.0000	0.0000	0.0300
qda	Quadratic Discriminant Analysis	0.4250	0.0000	0.5000	0.3000	0.3750	0.0000	0.0000	0.0230

Out[19]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

PyCaret library has been used to identify GaussianNB (Gaussian Naive Bayes) model as the best machine learning classification model for out data.

Gaussian Naive Bayes model details

In [20]:
```
# Visual representation of true positive, true negative, false positive, and false negative through confusion matrix
# confusion matrix plot:
plot_model(best, plot = 'confusion_matrix')
```



In [21]:
```
evaluate_model(best)
```

```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',), options=(('Pipeline Plot', 'pipelin…
```

In [23]:
```
# best model prediction
holdout_pred = predict_model(best)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

In [24]:
```
# Display of predictions tale
holdout_pred.head()
```

Out[24]:

	SRL_NO	PDF_NAME	PAGE_NUMBER	IL_HEADING	PL_HEADING	BOL_HEADING	CONTAINER_DETAILS	NO_OF_PCKGS	DESCRIPTION	QTY	PRICE	NET_AMOUNT	GROSS_WEIGHT	NET_WEIGHT	MEASUREMENT	IS_PL	IS_BOL	IS_INVOICE	predic
21	22	mcc-34page	11	False	False	False	False	False	True	True	True	True	False	False	False	0	0	1	
64	65	BOL extended	3	False	False	True	True	True	True	False	False	False	True	False	True	0	1	0	
28	29	mcc-34page	18	False	False	False	False	False	True	True	True	True	False	False	False	0	0	1	
48	49	inv + pkg + bl 1	4	False	False	False	False	False	False	False	False	False	False	False	False	0	0	0	
20	21	mcc-34page	10	False	False	False	False	False	True	True	True	True	False	False	False	0	0	1	

In [25]:
```
# copy data and drop Class variable
# NBC-2.csv is used for testing
new_data = pd.read_csv("NBC-2.csv")
# This GNB model only classifies whether data is invoice or not; this can be repeated for Packing List and Bill of Lading
new_data.drop('IS_INVOICE', axis=1, inplace=True)
new_data.head()
```

Out[25]:

	SRL_NO	PDF_NAME	PAGE_NUMBER	IL_HEADING	PL_HEADING	BOL_HEADING	CONTAINER_DETAILS	NO_OF_PCKGS	DESCRIPTION	QTY	PRICE	NET_AMOUNT	GROSS_WEIGHT	NET_WEIGHT	MEASUREMENT	IS_PL	IS_BOL
0	1	Invoice (Scanned)	1	True	True	False	True	True	True	True	True	True	False	False	False	0	0
1	2	Invoice (Scanned)	2	True	True	False	True	True	True	True	True	True	False	False	False	0	0
2	3	Invoice (Scanned)	3	True	True	False	True	True	True	True	True	True	False	False	False	0	0
3	4	Invoice (Scanned)	4	True	True	False	True	True	True	True	True	True	False	False	False	0	0
4	5	Invoice (Scanned)	5	True	True	False	True	True	True	True	True	True	False	False	False	0	0

In [27]:
```
# predict model on new_data
predictions = predict_model(best, data = new_data)
predictions.head()
```

Out[27]:

	PDF_NAME	PAGE_NUMBER	IL_HEADING	PL_HEADING	BOL_HEADING	CONTAINER_DETAILS	NO_OF_PCKGS	DESCRIPTION	QTY	PRICE	NET_AMOUNT	GROSS_WEIGHT	NET_WEIGHT	MEASUREMENT	IS_PL	IS_BOL	prediction_label	prediction_score
	Invoice (Scanned)	1	True	True	False	True	True	True	True	True	True	False	False	False	0	0	1	1.0
	Invoice (Scanned)	2	True	True	False	True	True	True	True	True	True	False	False	False	0	0	1	1.0
	Invoice (Scanned)	3	True	True	False	True	True	True	True	True	True	False	False	False	0	0	1	1.0
	Invoice (Scanned)	4	True	True	False	True	True	True	True	True	True	False	False	False	0	0	1	1.0
	Invoice (Scanned)	5	True	True	False	True	True	True	True	True	True	False	False	False	0	0	1	1.0

Above, it can be seen that the rows are correctly predicted as Invoice (prediction_score = 1), so Gaussian Naive Bayes is accurate for Invoice prediction (and for PL and BOL predictions).

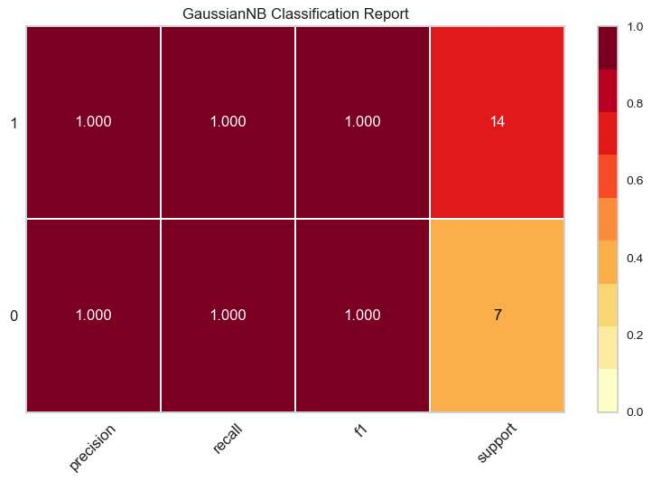
In [31]:
```
# save pipeline
save_model(best, 'project_pipeline')
# load pipeline
loaded_best_pipeline = load_model('project_pipeline')
loaded_best_pipeline
```

```
Transformation Pipeline and Model Successfully Saved
Transformation Pipeline and Model Successfully Loaded
```

Out[31]:
```
Pipeline(memory=FastMemory(location=C:\Users\RAMYAR~1\AppData\Local\Temp\joblib),
         steps=[('numerical_imputer',
                 TransformerWrapper(exclude=None,
                                    include=['SRL_NO', 'PAGE_NUMBER', 'IS_PL',
                                             'IS_BOL'],
                                    transformer=SimpleImputer(add_indicator=False,
                                                              copy=True,
                                                              fill_value=None,
                                                              keep_empty_features=False,
                                                              missing_values=nan,
                                                              strategy='mean'))),
                ('categorical_imputer',...
                                                              missing_values=nan,
                                                              strategy='most_frequent'))),
                ('onehot_encoding',
                 TransformerWrapper(exclude=None, include=['PDF_NAME'],
                                    transformer=OneHotEncoder(cols=['PDF_NAME'],
                                                              drop_invariant=False,
                                                              handle_missing='return_nan',
                                                              handle_unknown='value',
                                                              return_df=True,
                                                              use_cat_names=True,
                                                              verbose=0))),
                ('trained_model',
                 GaussianNB(priors=None, var_smoothing=1e-09))],
         verbose=False)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [32]:
```
# Visual representation of precision, recall, and f1 score through class report
plot_model(best, plot = 'class_report')
```



GAUSSIAN NAIVE BAYES MODEL: TRAINING

In [74]:
```
# importing scikit-Learn libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

3 MODELS (Invoice Identifier, Packing List Identifier, and Bill Of Lading Identifier): Three Gaussian Naive Bayes models are created to identify which category documents fall into. Each model will separately predict whether or not the pages in a document are Invoice, Packing list, or Bill of Lading.

DATA CLEANING AND MANIPULATION: The NBC-1.csv is further maniuplated to extract only useful columns to train the machine learning model with. Columns that involve the page no., name of PDF, whether the document is an invoice, packing list, or bill oflading are dropped depending on which model is being trained first.

Invoice GNB Model

In [75]:
```
# new data table reads NBC-1.csv and modifies it to match needs of training data by dropping certain columns
# X is the input data used for training and y is the data used to predict the outputs
# y is known as the Species column
data = pd.read_csv("NBC-1.csv")
data = data.drop("PAGE_NUMBER", axis=1)
data = data.drop("PDF_NAME", axis=1)
data = data.drop("IS_PL", axis=1)
data = data.drop("IS_BOL", axis=1)
X = data.drop("IS_INVOICE", axis=1)
y = data['IS_INVOICE']
data
```

Out[75]:

	SRL_NO	IL_HEADING	PL_HEADING	BOL_HEADING	CONTAINER_DETAILS	NO_OF_PCKGS	DESCRIPTION	QTY	PRICE	NET_AMOUNT	GROSS_WEIGHT	NET_WEIGHT	MEASUREMENT	IS_INVOICE
0	1	True	True	False	True	True	True	True	True	True	False	False	False	1
1	2	True	True	False	True	True	True	True	True	True	False	False	False	1
2	3	True	True	False	True	True	True	True	True	True	False	False	False	1
3	4	True	True	False	True	True	True	True	True	True	False	False	False	1
4	5	True	True	False	True	True	True	True	True	True	False	False	False	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
65	66	False	False	True	True	True	True	False	False	False	True	False	True	0
66	67	False	False	True	True	True	True	False	False	False	True	False	True	0
67	68	False	False	True	True	True	True	False	False	False	True	False	True	0
68	69	False	False	True	True	True	True	False	False	False	True	False	True	0
69	70	False	False	True	True	True	True	False	False	False	True	False	True	0

70% of the data is used for training and the remaining 30% is used for testing. These sets of data are chosen randomly to make them fair.

In [76]:
```
# Encoding (converting categorical data into numerical data) the Species column to get numerical class
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the data into training sets and testing sets
# 70% of the data is used for training and 30% for testing (randomly)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [77]:
```
# Gaussian Naive Bayes classifier
gnb = GaussianNB()
# Training the model using training data (X_train as the input and y_train as the output)
gnb.fit(X_train, y_train)
```

Out[77]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [78]:
```
# Using testing data (X_test) to make predictions (y_pred)
y_pred = gnb.predict(X_test)

# Accuracy of the model can be calculated as such:
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of invoice model after training 70% data and testing 30%: {accuracy}")
```

```
Accuracy of invoice model after training 70% data and testing 30%: 1.0
```

The accuracy of the model is 100%

In [79]:
```
# Here is an example of testing data
print(X_test.head())
```

```
    SRL_NO  IL_HEADING  PL_HEADING  BOL_HEADING  CONTAINER_DETAILS  \
22      23       False       False        False              False
0        1        True        True        False               True
49      50        True       False        False              False
4        5        True        True        False               True
54      55       False        True        False              False

    NO_OF_PCKGS  DESCRIPTION   QTY  PRICE  NET_AMOUNT  GROSS_WEIGHT  \
22        False         True  True   True        True         False
0          True         True  True   True        True         False
49        False         True  True   True        True         False
4          True         True  True   True        True         False
54         True         True  True  False       False          True

    NET_WEIGHT  MEASUREMENT
22       False        False
0        False        False
49       False        False
4        False        False
54        True         True
```

In [80]:
```
# Here is an example of the output prediction where 1 means the page is an Invoice and 0 means it is not an Invoice
print(y_pred[0:5])
```

```
[1 1 1 1 0]
```

Now that we have trained the Invoice model, we can do the same for the Packing List and Bill of Lading models

Packing List GNB Model

In [81]:
```
# the same process as in invoice is repeated but with different columns according the the needs of the training data
data = pd.read_csv("NBC-1.csv")
data = data.drop("PAGE_NUMBER", axis=1)
data = data.drop("PDF_NAME", axis=1)
data = data.drop("IS_INVOICE", axis=1)
data = data.drop("IS_BOL", axis=1)
X = data.drop("IS_PL", axis=1)
y = data['IS_PL']
data
```

Out[81]:

	SRL_NO	IL_HEADING	PL_HEADING	BOL_HEADING	CONTAINER_DETAILS	NO_OF_PCKGS	DESCRIPTION	QTY	PRICE	NET_AMOUNT	GROSS_WEIGHT	NET_WEIGHT	MEASUREMENT	IS_PL
0	1	True	True	False	True	True	True	True	True	True	False	False	False	0
1	2	True	True	False	True	True	True	True	True	True	False	False	False	0
2	3	True	True	False	True	True	True	True	True	True	False	False	False	0
3	4	True	True	False	True	True	True	True	True	True	False	False	False	0
4	5	True	True	False	True	True	True	True	True	True	False	False	False	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
65	66	False	False	True	True	True	True	False	False	False	True	False	True	0
66	67	False	False	True	True	True	True	False	False	False	True	False	True	0
67	68	False	False	True	True	True	True	False	False	False	True	False	True	0
68	69	False	False	True	True	True	True	False	False	False	True	False	True	0
69	70	False	False	True	True	True	True	False	False	False	True	False	True	0

70 rows × 14 columns

In [82]:
```
# Encoding (converting categorical data into numerical data) the Species column to get numerical class
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the data into training sets and testing sets
# 70% of the data is used for training and 30% for testing (randomly)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [83]:
```
# Gaussian Naive Bayes classifier
gnb = GaussianNB()
# Training the model using training data (X_train as the input and y_train as the output)
gnb.fit(X_train, y_train)
```

Out[83]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [84]:
```
# Using testing data (X_test) to make predictions (y_pred)
y_pred = gnb.predict(X_test)

# Accuracy of the model can be calculated as such:
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of PL model after training 70% data and testing 30%: {accuracy}")
```

```
Accuracy of PL model after training 70% data and testing 30%: 1.0
```

The accuracy of the model is 100%

Bill Of Lading GNB Model

In [85]:
```
# the same process as in invoice is repeated but with different columns according the the needs of the training data
data = pd.read_csv("NBC-1.csv")
data = data.drop("PAGE_NUMBER", axis=1)
data = data.drop("PDF_NAME", axis=1)
data = data.drop("IS_INVOICE", axis=1)
data = data.drop("IS_PL", axis=1)
X = data.drop("IS_BOL", axis=1)
y = data['IS_BOL']
data
```

Out[85]:

	SRL_NO	IL_HEADING	PL_HEADING	BOL_HEADING	CONTAINER_DETAILS	NO_OF_PCKGS	DESCRIPTION	QTY	PRICE	NET_AMOUNT	GROSS_WEIGHT	NET_WEIGHT	MEASUREMENT	IS_BOL
0	1	True	True	False	True	True	True	True	True	True	False	False	False	0
1	2	True	True	False	True	True	True	True	True	True	False	False	False	0
2	3	True	True	False	True	True	True	True	True	True	False	False	False	0
3	4	True	True	False	True	True	True	True	True	True	False	False	False	0
4	5	True	True	False	True	True	True	True	True	True	False	False	False	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
65	66	False	False	True	True	True	True	False	False	False	True	False	True	1
66	67	False	False	True	True	True	True	False	False	False	True	False	True	1
67	68	False	False	True	True	True	True	False	False	False	True	False	True	1
68	69	False	False	True	True	True	True	False	False	False	True	False	True	1
69	70	False	False	True	True	True	True	False	False	False	True	False	True	1

70 rows × 14 columns

In [86]:
```
# Encoding (converting categorical data into numerical data) the Species column to get numerical class
le = LabelEncoder()
y = le.fit_transform(y)

# Splitting the data into training sets and testing sets
# 70% of the data is used for training and 30% for testing (randomly)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [87]:
```
# Gaussian Naive Bayes classifier
gnb = GaussianNB()
# Training the model using training data (X_train as the input and y_train as the output)
gnb.fit(X_train, y_train)
```

Out[87]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [88]:
```
# Using testing data (X_test) to make predictions (y_pred)
y_pred = gnb.predict(X_test)

# Accuracy of the model can be calculated as such:
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of BOL model after training 70% data and testing 30%: {accuracy}")
```

```
Accuracy of BOL model after training 70% data and testing 30%: 1.0
```

The accuracy of the model is 100%

Now that our models are created (trained and tested), we can classify documents into the 4 categories by creating a csv file called input.csv that is a table used as the input for the models. The file will use contain all the columns mentioned previously and an API (Nanonets OCR Package) will be used to identify if certain features are present in a document and fill it in the table.

OUTPUTS AND PREDICTIONS

API KEY: 767fdbe0-2d24-11ef-9844-3a3797bf48b8

OR if key doesn't work: Get your free API Key -

1. Go to https://app.nanonets.com/#/signup?&utm_source=wrapper (https://app.nanonets.com/#/signup?&utm_source=wrapper)
2. On your Nanonets account, Go to Account Info -> API Keys. Generate your free API Key and copy it
3. Use it to authenticate this Nanonets OCR Wrapper.

Then, enter the name of the PDF/Document that contains the pages required to be classified. Ensure that the document is in the same folder of the program that is being run.

In [101]:
```
# Nanonets OCR Package
from nanonets import NANONETSOCR
model = NANONETSOCR()
# token = input("Enter API key (767fdbe0-2d24-11ef-9844-3a3797bf48b8) or check documentation: \n")
model.set_token(token)
# input_file = input("Enter PDF path to be converted:\n")
```

Using the OCR package, we can generate a lists of words present in the page:

In [102]:
```
boxes = model.convert_to_boxes(input_file)
output_list = [box for box in boxes]
text_list = []
for list1 in output_list:
    text_list.append([dictionary['text'] for dictionary in list1])
final_list_of_words = []
for l in text_list:
    for word in l:
        final_list_of_words.append(word)
```

Now, we can create the input.csv file by checking the list of words recognized by the OCR package and filling in the rows according to certain conditions:

In [103]:

```
# creatinf field names for input.csv file
field_names= ["SRL_NO", "IL_HEADING", "PL_HEADING", "BOL_HEADING",
              "CONTAINER_DETAILS", "NO_OF_PCKGS", "DESCRIPTION",
              "QTY", "PRICE", "NET_AMOUNT", "GROSS_WEIGHT",
              "NET_WEIGHT", "MEASUREMENT"]
# making csv
field_names= ["SRL_NO", "IL_HEADING", "PL_HEADING", "BOL_HEADING",
              "CONTAINER_DETAILS", "NO_OF_PCKGS", "DESCRIPTION",
              "QTY", "PRICE", "NET_AMOUNT", "GROSS_WEIGHT",
              "NET_WEIGHT", "MEASUREMENT"]
csv_list = []
i = 1
for page in text_list:
    ilh = "FALSE"
    plh = "FALSE"
    bolh = "FALSE"
    # checking packing list header
    if "PACKING" in page:
        plh = "TRUE"
    elif "packing" in page:
        plh = "TRUE"
    if "INVOICE/PACKING" in page or "invoice/packing" in page:
        ilh = "TRUE"
        plh = "TRUE"
    # checking invoice header
    if "INVOICE" in page:
        ilh = "TRUE"
    elif "invoice" in page:
        ilh = "TRUE"
    # checking bill of lading header
    if "BILL" in page or "bill" in page or "Bill" in page:
        if "OF" in page or "of" in page or "Of" in page:
            if "LADING" in page or "lading" in page or "Lading" in page:
                bolh = "TRUE"
    # checking container details
    if "Marks" in page or "MARKS" in page or "marks" in page or "mark" in page or "Mark" in page:
        cd = "TRUE"
    elif "Marks" in page and "&" in page and "No.s" in page:
        cd = "TRUE"
    elif "Marks" in page and "&" in page and "Numbers" in page:
        cd = "TRUE"
    elif "Marks" in page and "and" in page and "Numbers" in page:
        cd = "TRUE"
    else:
        cd = "FALSE"

    # checking number of packages
    if "No." in page or "Number" in page or "NO." in page:
        if "Packages" in page or "packages" in page or "Pkgs." in page or "pkgs." in page or "PACKAGES" in page:
            nop = "TRUE"
        else:
            nop = "TRUE"
    else:
        nop = "FALSE"

    # checking description
    if "Description" in page or "description" in page or "DESCRIPTION" in page:
        des = "TRUE"
    else:
        des = "FALSE"

    #checking quantity
    if "Quantity" in page or "quantity" in page or "Qty." in page or "qty." in page or "Qty" in page or "qty" in page or "QUANTITY" in page:
        qty = "TRUE"
    else:
        qty = "FALSE"

    # checking price
    price = "FALSE"
    if any(word in s for s in page for word in ["Rate", "rate", "Price", "price", "PRICE"]):
        price = "TRUE"

    # checking net amount
    neta = "FALSE"
    if any(word in s for s in page for word in ["Amount", "amount", "Amt.", "amt.", "AMOUNT"]):
        neta = "TRUE"

    # checking gross weight
    gw = "FALSE"
    if any(word in s for s in page for word in ["gross", "Gross", "G.W"]):
        if any(word in s for s in page for word in ["weight", "Weight", "WEIGHT", "G.W"]):
            gw = "TRUE"
    elif "G.W." in page or "g.w." in page or "GW/NW" in page or "GW" in page or "Gross" in page or "gross" in page or "GROSS" in page:
        gw = "TRUE"
    elif "Gross" in page and "Weight" in page:
        gw = "TRUE"
    elif "gross" in page and "weight" in page:
        gw = "TRUE"

    # checking net weight
    nw = "FALSE"
    if any(word in s for s in page for word in ["N.W", "NW", "net", "Net", "n.w.", "N.W.", "NET"]):
        if any(word in s for s in page for word in ["N.W", "NW", "weight", "Weight", "WEIGHT", "n.w.", "N.W."]):
            nw = "TRUE"
    elif any(word in page for word in ["N.", "n."]):
        if any(word in s for s in page for word in [".W", ".w"]):
            nw = "TRUE"
    elif any(word in s for s in page for word in ["Weight", "weight", "WEIGHT"]):
        if any(word in s for s in page for word in ["net", "Net", "NET"]):
            nw = "TRUE"
    elif "N.W." in page or "n.w." in page or "GW/NW" in page or "NW" in page or "Net" in page or "net" in page:
        nw = "TRUE"
    elif "Net" in page and "Weight" in page:
        nw = "TRUE"
    elif "net" in page and "weight" in page:
        nw = "TRUE"

    # checking measurement
    if "Measurement" in page or "MEASUREMENT" in page or "measurement" in page:
        m = "TRUE"
    else:
        m = "FALSE"
    csv_list.append({"SRL_NO": i,
                     "IL_HEADING": ilh,
                     "PL_HEADING": plh,
                     "BOL_HEADING": bolh,
                     "CONTAINER_DETAILS": cd,
                     "NO_OF_PCKGS": nop,
                     "DESCRIPTION": des,
                     "QTY": qty,
                     "PRICE": price,
                     "NET_AMOUNT": neta,
                     "GROSS_WEIGHT": gw,
                     "NET_WEIGHT": nw,
                     "MEASUREMENT": m})
    i += 1
```

In [104]:

```
# writing the csv
import csv
with open("input.csv", "w") as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames=field_names)
    writer.writeheader()
    writer.writerows(csv_list)
```

Now that a file called input.csv is written by indetifying fields present in the input document, it can be ran through the GNB models that have been created to predict the classes for all pages of the document.

In [105]:
```
new_test = pd.read_csv("input.csv")
new_test.dropna(inplace = True)
new_invoice_prediction = gnb.predict(new_test)
new_pl_prediction = gnb.predict(new_test)
new_bol_prediction = gnb.predict(new_test)
```

These new_doctype_prediction variables are arrays that contain 1s and 0s depending on whether the page satisfies the model or not

Now, that we have predicted what kind of pages the document can be classified into, we can print the type of each page individually to give an overview of the document.

In [112]:
```
'''
j=0
while j < len(new_invoice_prediction):
    if new_invoice_prediction[j] == 1:
         #print("Invoice")
    elif new_pl_prediction[j] == 1:
        #print("Packing List")
    elif new_bol_prediction[j] == 1:
        #print("Bill of Lading")
    else:
         #print("Other")
    j += 1
'''
```

Out[112]:
```
'\nj=0\nwhile j < len(new_invoice_prediction):\n    if new_invoice_prediction[j] == 1:\n         #print("Invoice")\n    elif new_pl_prediction[j] == 1:\n        #print("Packing List")\n    elif new_bol_pre
diction[j] == 1:\n        #print("Bill of Lading")\n    else:\n         #print("Other")\n    j += 1\n'
```

This program can be run through the python file as well

Author: Ramya Rajaram

References:

OCR Package: https://github.com/NanoNets/ocr-python/tree/main (https://github.com/NanoNets/ocr-python/tree/main)

PyCaret Binary Classification Tutorial: https://github.com/pycaret/pycaret/blob/master/tutorials/Tutorial%20-%20Binary%20Classification.ipynb (https://github.com/pycaret/pycaret/blob/master/tutorials/Tutorial%20-%20Binary%20Classification.ipynb)

GNB model: https://scikit-learn.org/0.15/modules/generated/sklearn.naive_bayes.GaussianNB.html (https://scikit-learn.org/0.15/modules/generated/sklearn.naive_bayes.GaussianNB.html)

In [ ]:
```
```