AIRLINE DELAY PREDICTION

BIG DATA PROJECT

PHASE II

Group - I

Pooja Chandwadkar
Decision Tree
Made Use of Hive & Rapidminer

Jyoti Kumari
Naive Bayes
Made Use of Hive & Rapidminer

Leena Bhirud
Random Forest
Made Use of Hive & Rapidminer

Mohammed Zeeshan Ali
Support Vector Machine
Made Use of Python

Raviraj Ahire
K-Nearest Neighbour
Made Use of Python & Rapidminer

Deval Kaku
Logistic Regression
Made Use of Python & Rapidminer

AGENDA

1. Problem Statement
2. Data Extraction
3. Data Visualization
4. Models
    - Random Forest Classifier
    - Decision Tree
    - Naive Bayes
    - Support Vector Machine(SVM)
    - Logistic Regression
    - KNN Algorithm
5. Findings
6. Prediction
7. Conclusion

PROBLEM STATEMENT

- Airline on time data is fetched from Harvard Dataverse website for year 1993, 2000, 2005, 2006,2007,2008 to calculate the flight delays.
- We used On Premise and Cloud Platform to perform ETL(Extract Transform Load)
- We used ‘Rapidminer’ and ‘Python’ for building various predictive analytical models.

AIRLINE DATA EXTRACTION USING PYTHON

```
#Data Extraction
File = ['1993.csv','2000.csv','2005.csv','2006.csv','2007.csv','2008.csv']
File_location= "/Users/zee/Desktop/Big Data project/"
Extension = '_new.csv'
Airline_delay = pd.DataFrame()

for Filename in File:
    Import_data = File_location + Filename       # Fetching the data of the years
    data = pd.read_csv(Import_data)              # Loading the data into the platform
    random = data.sample(1000,random_state=42)   # random selection of data 1000 records from each year

    save_file = File_location + Filename + Extension
    random.to_csv(save_file)

    Airline_delay = pd.concat([Airline_delay,random])   #combining the loaded data with years
    Airline_delay.to_csv(r'/Users/zee/Desktop/Big Data project/Airline_delay.csv',index=False))
```

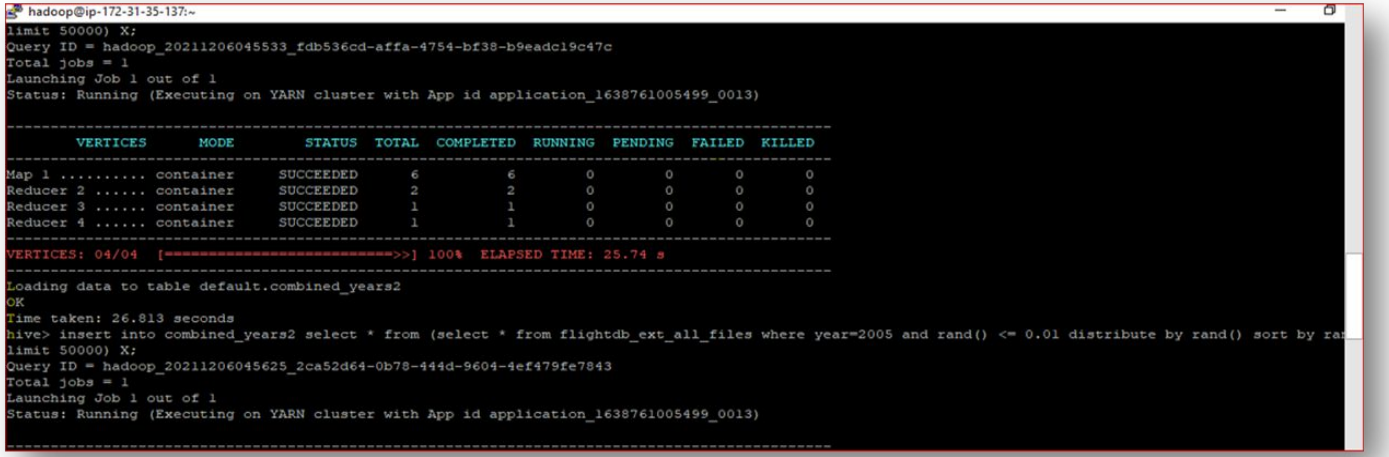
	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	...	TaxiIn	TaxiOut	Cancelled	CancellationCode	Diverted	CarrierDelay	WeatherDelay	NASDelay	SecurityDelay	LateAircraftDelay
4209789	1993	10	7	4	730.0	730	1124.0	1119	CO	352	...	NaN	NaN	0	NaN	0	NaN	NaN	NaN	NaN	NaN
2377031	1993	6	25	5	1430.0	1430	1523.0	1531	DL	1837	...	NaN	NaN	0	NaN	0	NaN	NaN	NaN	NaN	NaN
120245	1993	1	19	2	2043.0	2045	2319.0	2339	NW	894	...	NaN	NaN	0	NaN	0	NaN	NaN	NaN	NaN	NaN
4854736	1993	12	16	4	1224.0	1225	1329.0	1333	US	81	...	NaN	NaN	0	NaN	0	NaN	NaN	NaN	NaN	NaN
4115444	1993	10	1	5	1644.0	1645	1902.0	1836	NW	107	...	NaN	NaN	0	NaN	0	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3731280	2007	7	10	2	744.0	740	836.0	825	WN	2653	...	5.0	17.0	0	NaN	0	0.0	0.0	0.0	0.0	0.0
1950791	2007	4	27	5	1604.0	1335	1835.0	1535	XE	2807	...	8.0	51.0	0	NaN	0	0.0	0.0	180.0	0.0	0.0
420454	2007	1	27	6	657.0	700	833.0	835	MQ	4431	...	25.0	8.0	0	NaN	0	0.0	0.0	0.0	0.0	0.0
4737383	2007	8	9	4	1242.0	1246	2034.0	2030	FL	16	...	13.0	21.0	0	NaN	0	0.0	0.0	0.0	0.0	0.0
6918674	2007	12	27	4	858.0	855	1113.0	1125	WN	1928	...	4.0	9.0	0	NaN	0	0.0	0.0	0.0	0.0	0.0

6000 rows × 29 columns

AIRLINE DATA EXTRACTION USING HIVE

```
[hadoop@ip-172-31-35-137 ~]$ hdfs dfs -ls /user/hadoop/examdata/
Found 6 items
-rw-r--r--   1 hadoop hdfsadmingroup  501558665 2021-12-06 03:37 /user/hadoop/examdata/1993.csv
-rw-r--r--   1 hadoop hdfsadmingroup  501558665 2021-12-06 03:37 /user/hadoop/examdata/2000.csv
-rw-r--r--   1 hadoop hdfsadmingroup  501558665 2021-12-06 03:37 /user/hadoop/examdata/2005.csv
-rw-r--r--   1 hadoop hdfsadmingroup  501558665 2021-12-06 03:37 /user/hadoop/examdata/2006.csv
-rw-r--r--   1 hadoop hdfsadmingroup  501558665 2021-12-06 03:37 /user/hadoop/examdata/2007.csv
-rw-r--r--   1 hadoop hdfsadmingroup  501558665 2021-12-06 03:38 /user/hadoop/examdata/2008.csv
[hadoop@ip-172-31-35-137 ~]$
```

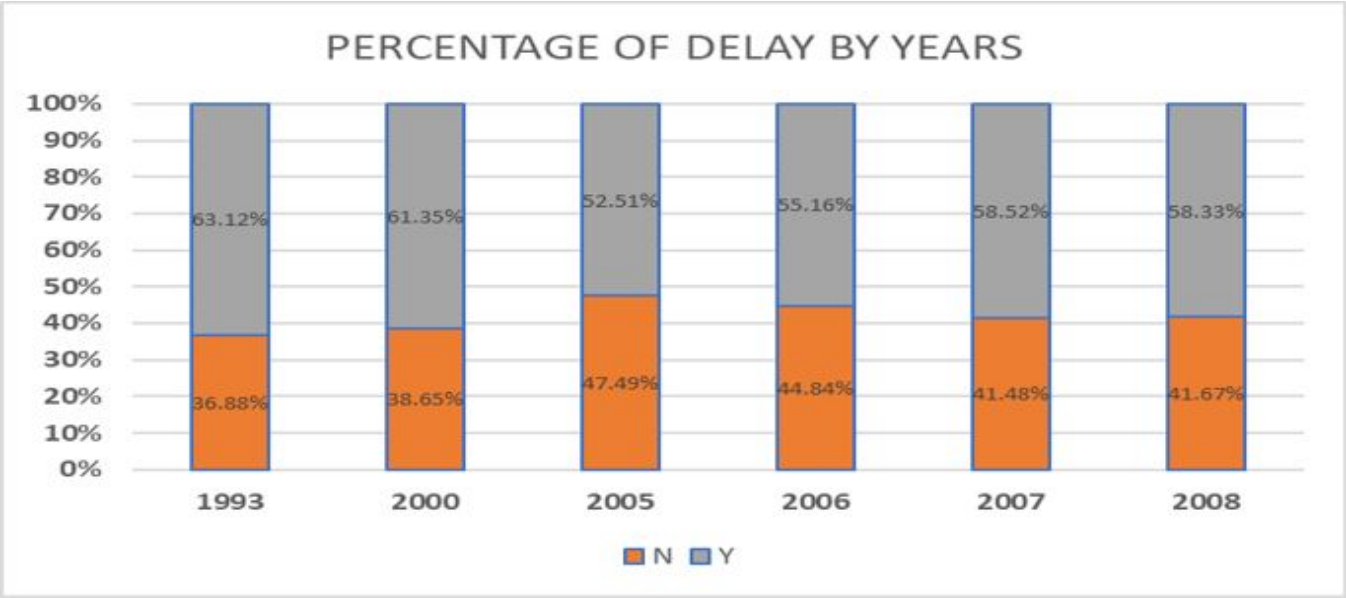
```
hive> select distinct year from flightdb_ext_all_files limit 20;
Query ID = hadoop_20211206040722_739623bc-feaa-4f71-ae31-81158315a9dc
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1638761005499_0006)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........  container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ......  container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 10.64 s
----------------------------------------------------------------------------------------------
OK
1993
2000
2005
2006
2007
2008
Time taken: 12.757 seconds, Fetched: 6 row(s)
hive>
```



- Load randomize data for all year
- Create a table CombinedYears and Add a new column “ Delayed”
- Save .csv file to S3 bucket in EC2

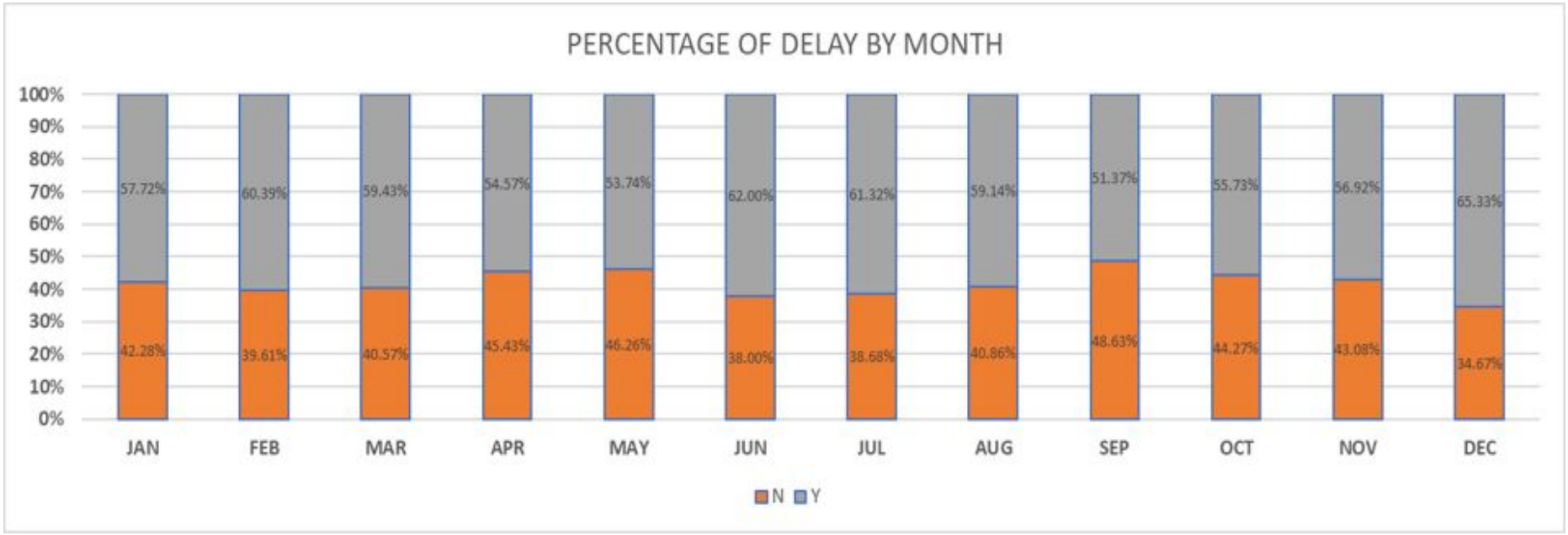
DATA VISUALIZATION !

PERCENTAGE OF DELAY BY YEARS



DELAY RATE HAS BEEN DECREASING SINCE 2005, COMPARED BEFORE 2005

PERCENTAGE OF DELAY BY MONTHS



DECEMBER HAS THE HIGHEST DELAY CHANCE FOLLOWED BY JUNE & JULY

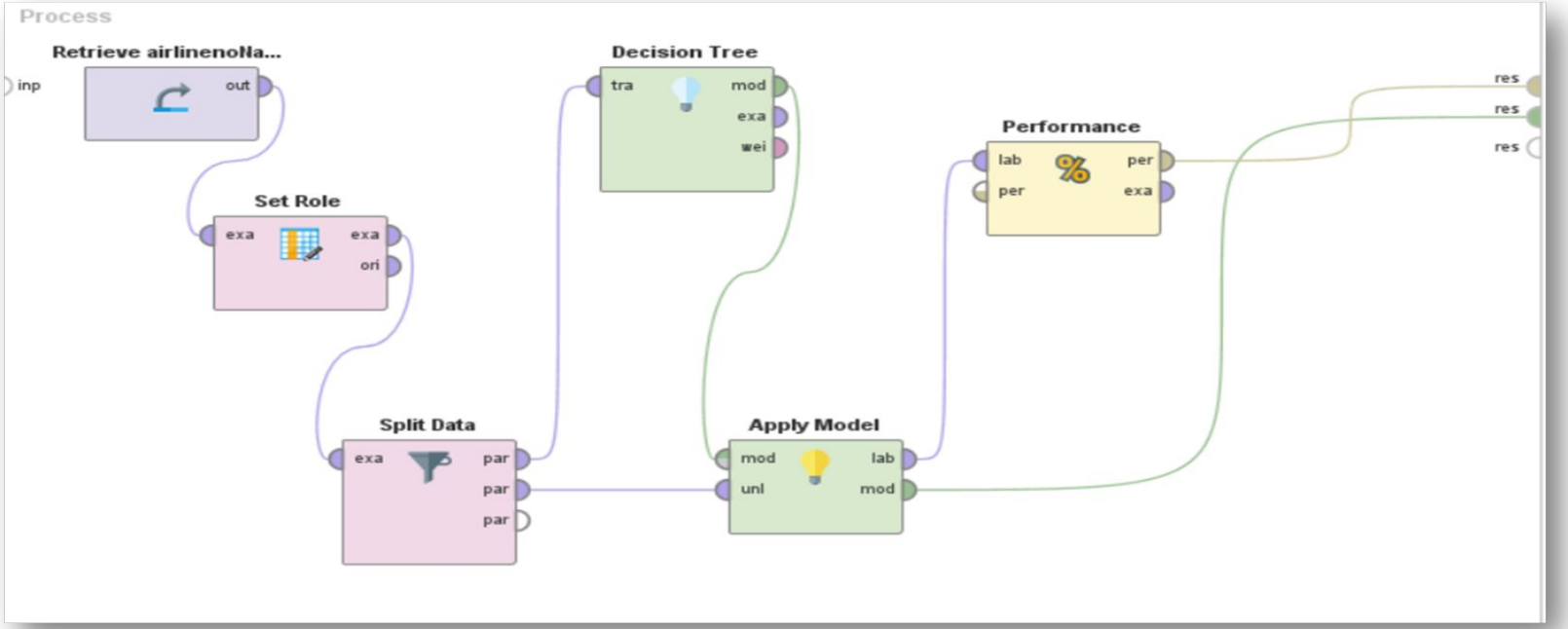
DECISION TREE !

WHAT IS DECISION TREE?

- Graphical representation of various alternative solution to the given problem

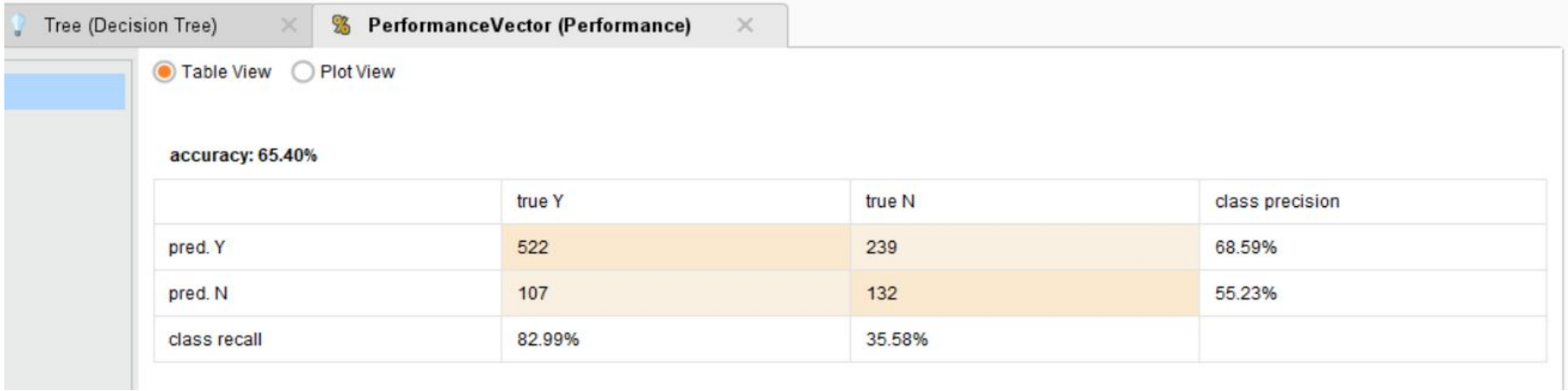
- Allow us to analyze all possible consequences

DESIGN



TARGET COLUMN : LABEL : DELAYED

TRAINING MODEL - PERFORMANCE



	true Y	true N	class precision
pred. Y	522	239	68.59%
pred. N	107	132	55.23%
class recall	82.99%	35.58%	

- Classification error: 37.2%
- Accuracy: 65.40%
  - Predicted Y (68.59%)
    - 522 Predicted true Y
    - 239 Predicted true N
  - Predicted N (55.23%)
    - 107 Predicted true Y
    - 132 Predicted true N

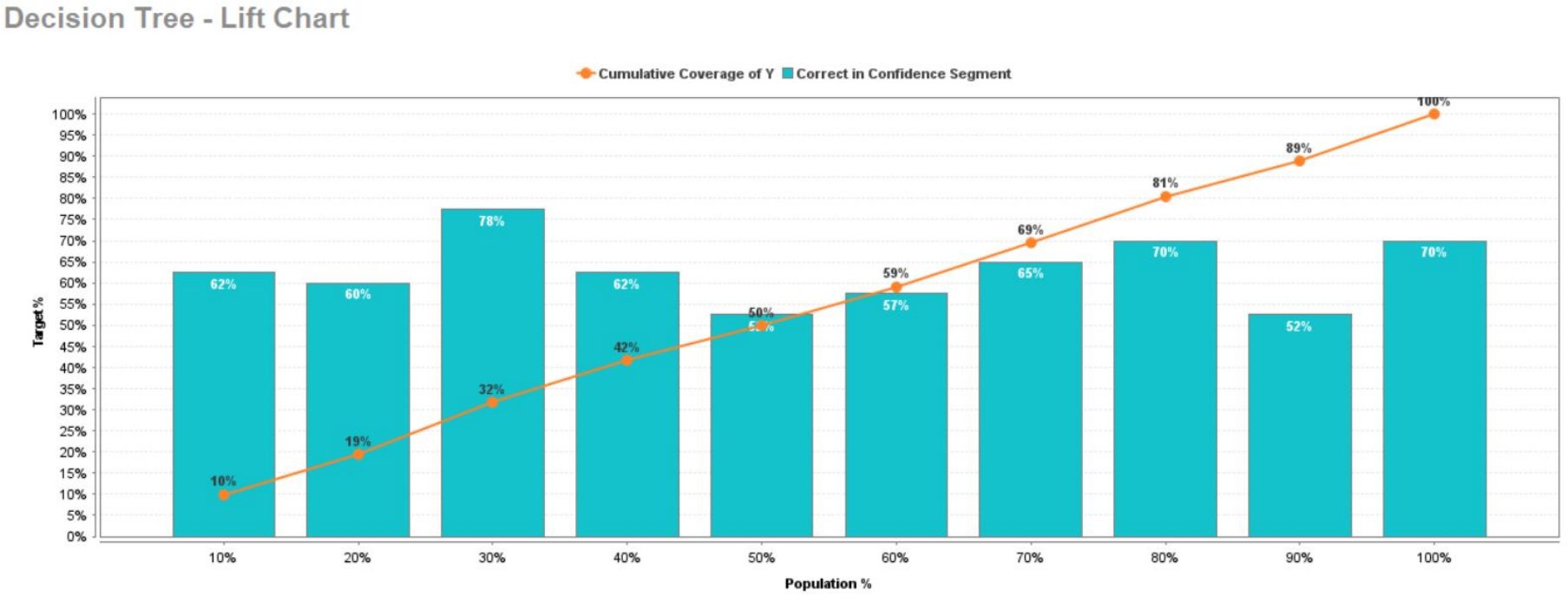
DECISION TREE - PERFORMANCE - AUTOMODEL

Decision Tree - Performance

Performances

Criterion	Value	Standard Deviation
Accuracy	62.2%	± 2.8%
Classification Error	37.8%	± 2.8%
AUC	50.0%	± 0.0%
Precision	62.2%	± 2.8%
Recall	100.0%	± 0.0%
F Measure	76.7%	± 2.1%
Sensitivity	100.0%	± 0.0%
Specificity	0.0%	± 0.0%

DECISION TREE - LIFT CHART



DECISION TREE - PREDICTIONS

Decision Tree - Predictions

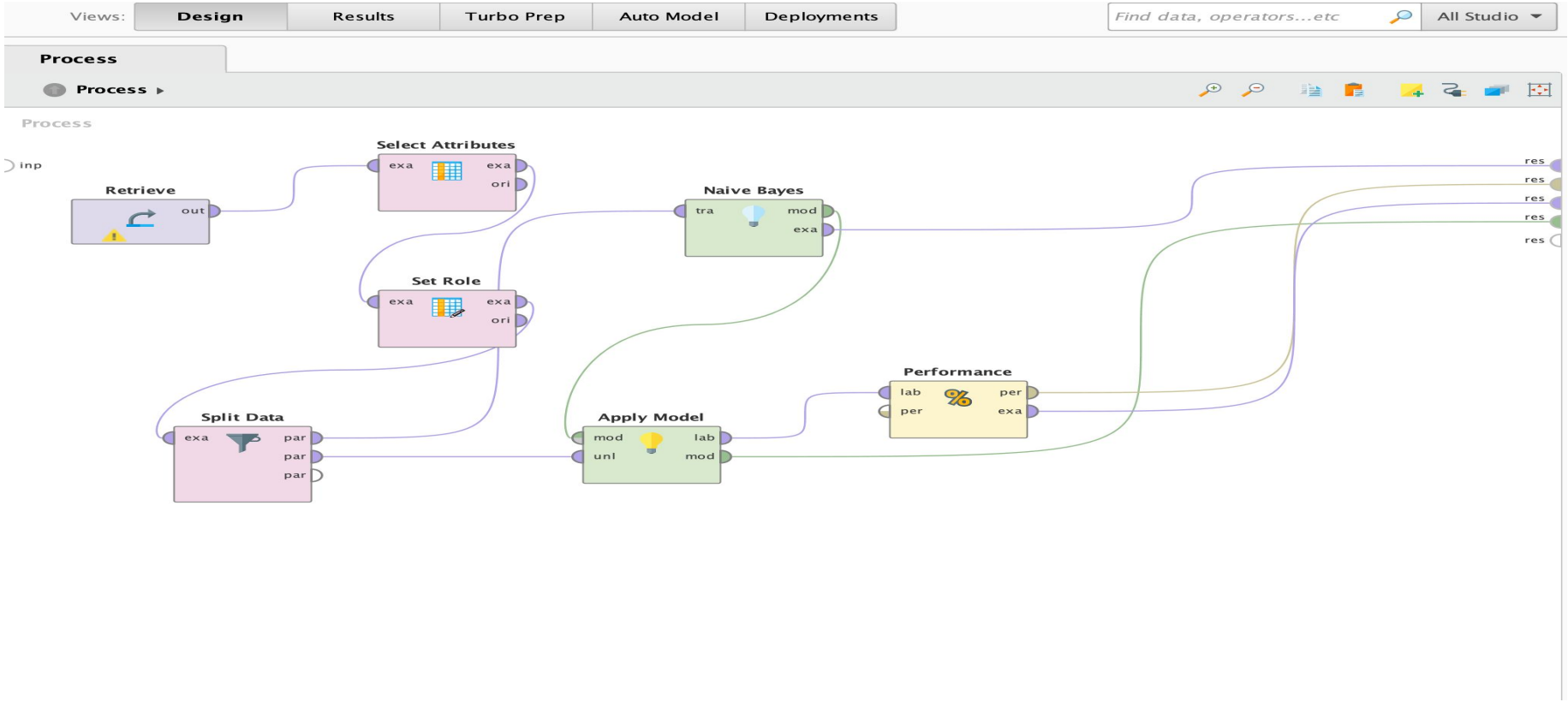
Row No.	Delayed	prediction(D...	confidence(N)	confidence(Y)	cost	UniqueCarrier
1	Y	Y	0.356	0.644	0.288	DL
2	Y	Y	0.356	0.644	0.288	UA
3	Y	Y	0.356	0.644	0.288	UA
4	Y	Y	0.356	0.644	0.288	AA
5	Y	Y	0.356	0.644	0.288	TW
6	Y	Y	0.356	0.644	0.288	AA
7	Y	Y	0.356	0.644	0.288	US
8	Y	Y	0.356	0.644	0.288	WN
9	N	Y	0.356	0.644	0.288	WN
10	N	Y	0.356	0.644	0.288	DL
11	N	Y	0.356	0.644	0.288	TW
12	Y	Y	0.356	0.644	0.288	WN
13	Y	Y	0.356	0.644	0.288	AA

NAIVE BAYES !

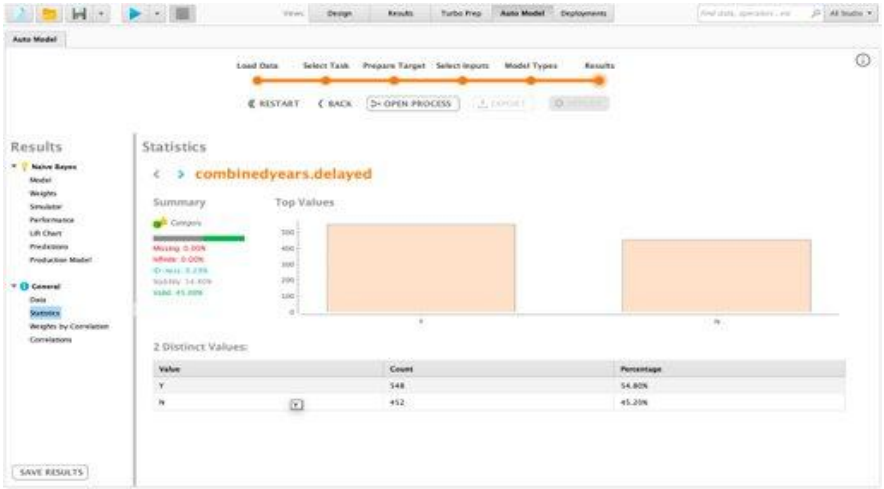
WHAT IS NAIVE BAYES?

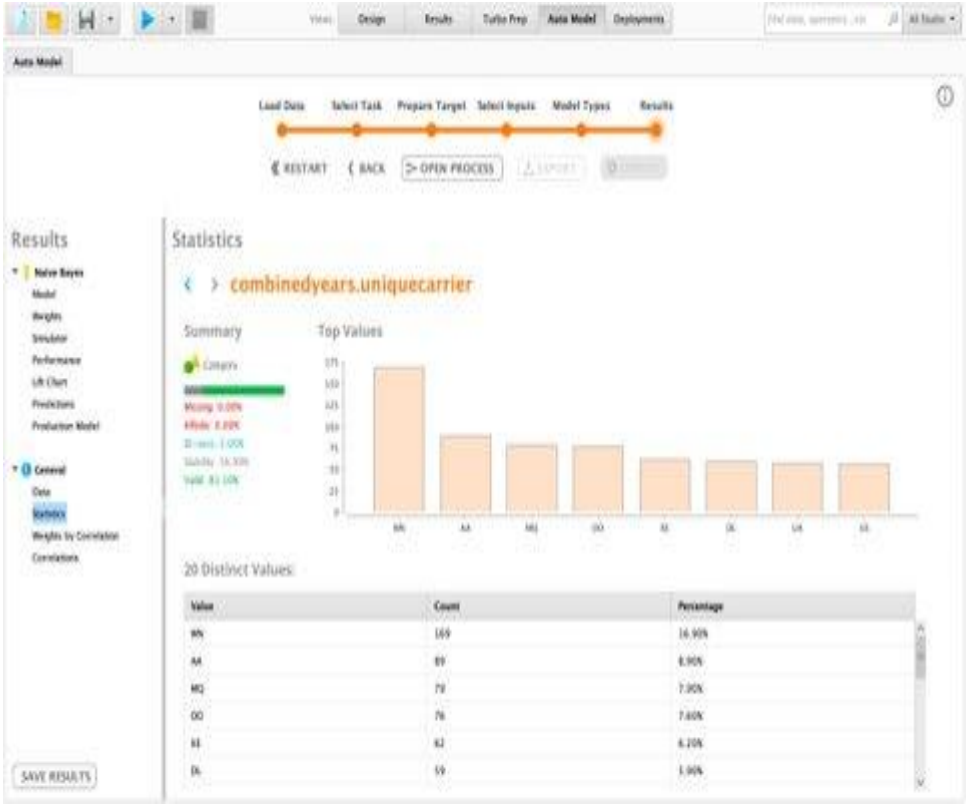
- A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task.
- Naive Bayes algorithms are mostly used in sentiment analysis, spam filtering, recommendation systems

Design

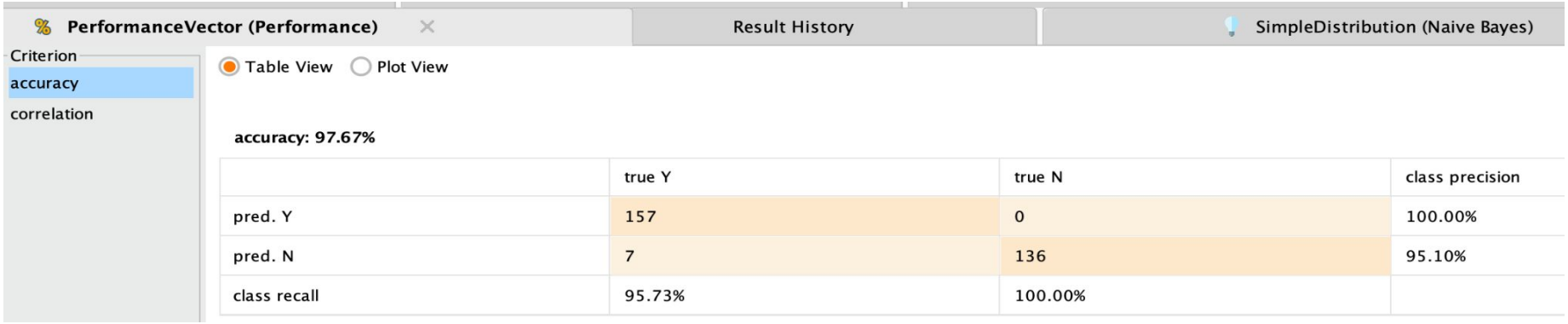

Views:
Design
Results
Turbo Prep
Auto Model
Deployments
Find data, operators...etc
All Studio
Process
Process
Process
inp
Retrieve
out
Select Attributes
exa
exa
ori
Set Role
exa
exa
ori
Split Data
exa
par
par
par
Naive Bayes
tra
mod
exa
Apply Model
mod
lab
unl
mod
Performance
lab
per
per
exa
res
res
res
res
res

NAIVE BAYES


Auto Model
Load Data
Select Task
Prepare Target
Select Inputs
Model Types
Results
RESTART
BACK
OPEN PROCESS
Results
Statistics
combinedyears.delayed
Summary
Top Values
2 Distinct Values:
SAVE RESULTS


Auto Model
Load Data
Select Task
Prepare Target
Select Inputs
Model Types
Results
RESTART
BACK
OPEN PROCESS
Results
Statistics
combinedyears.uniquecarrier
Summary
Top Values
20 Distinct Values:
SAVE RESULTS

NAIVE BAYES - PERFORMANCE

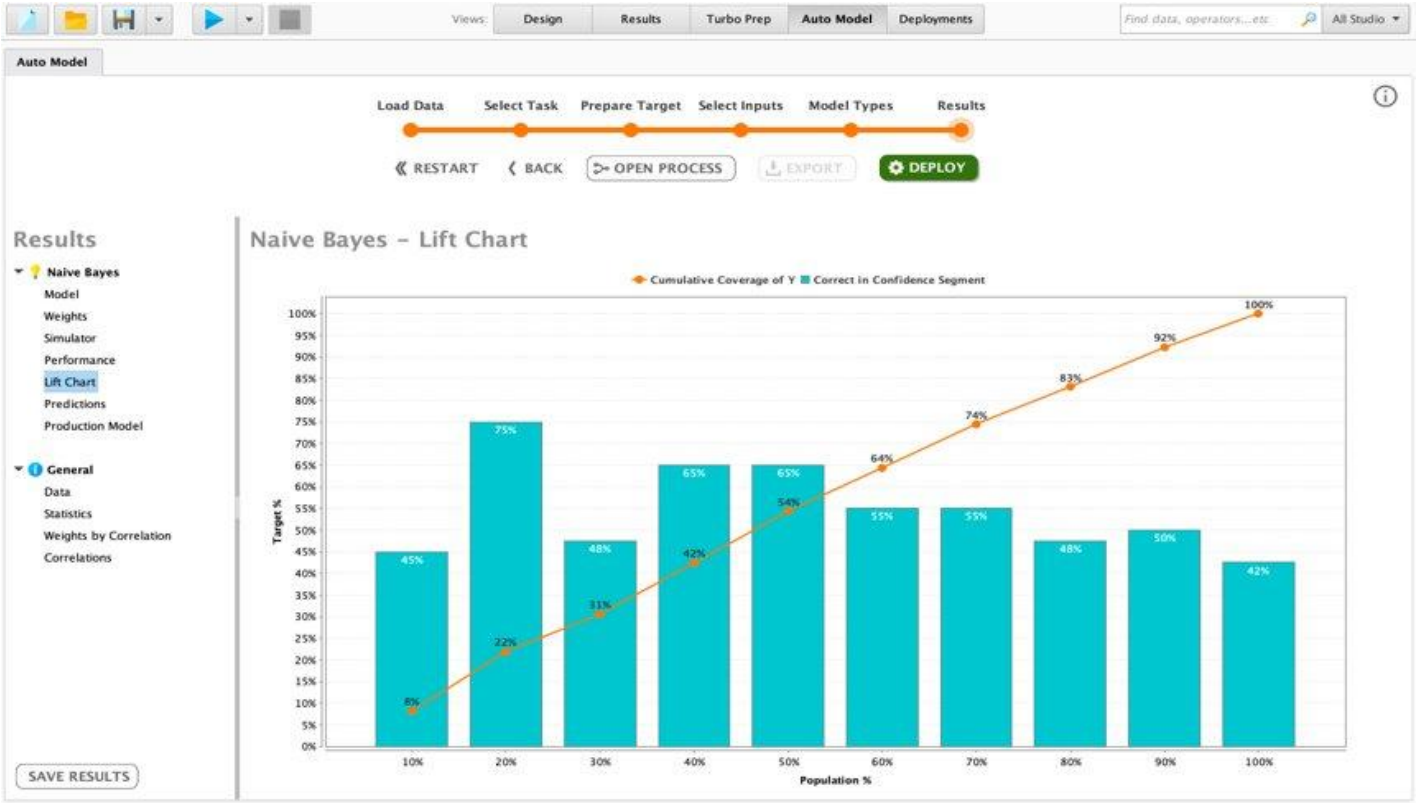


	true Y	true N	class precision
pred. Y	157	0	100.00%
pred. N	7	136	95.10%
class recall	95.73%	100.00%	

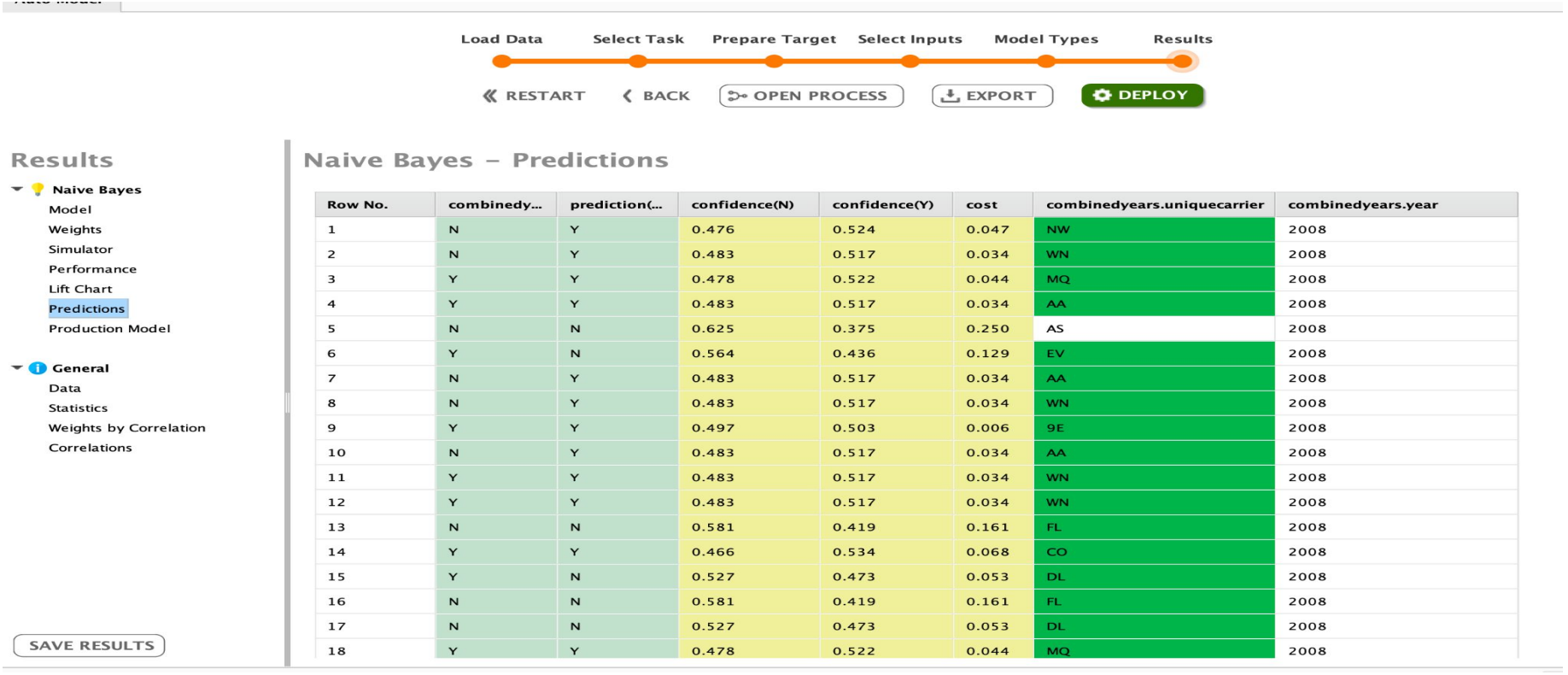
- Classification error: 45.5%
- Accuracy: 97.67%
  - Predicted Y (100%)
    - 157 Predicted true Y
    - 0 Predicted true N
  - Predicted N (95.10%)
    - 136 Predicted true N
    - 7 Predicted true Y

Correlation : 1.000

NAIVE BAYES - LIFT CHART



NAIVE BAYES - PREDICTION



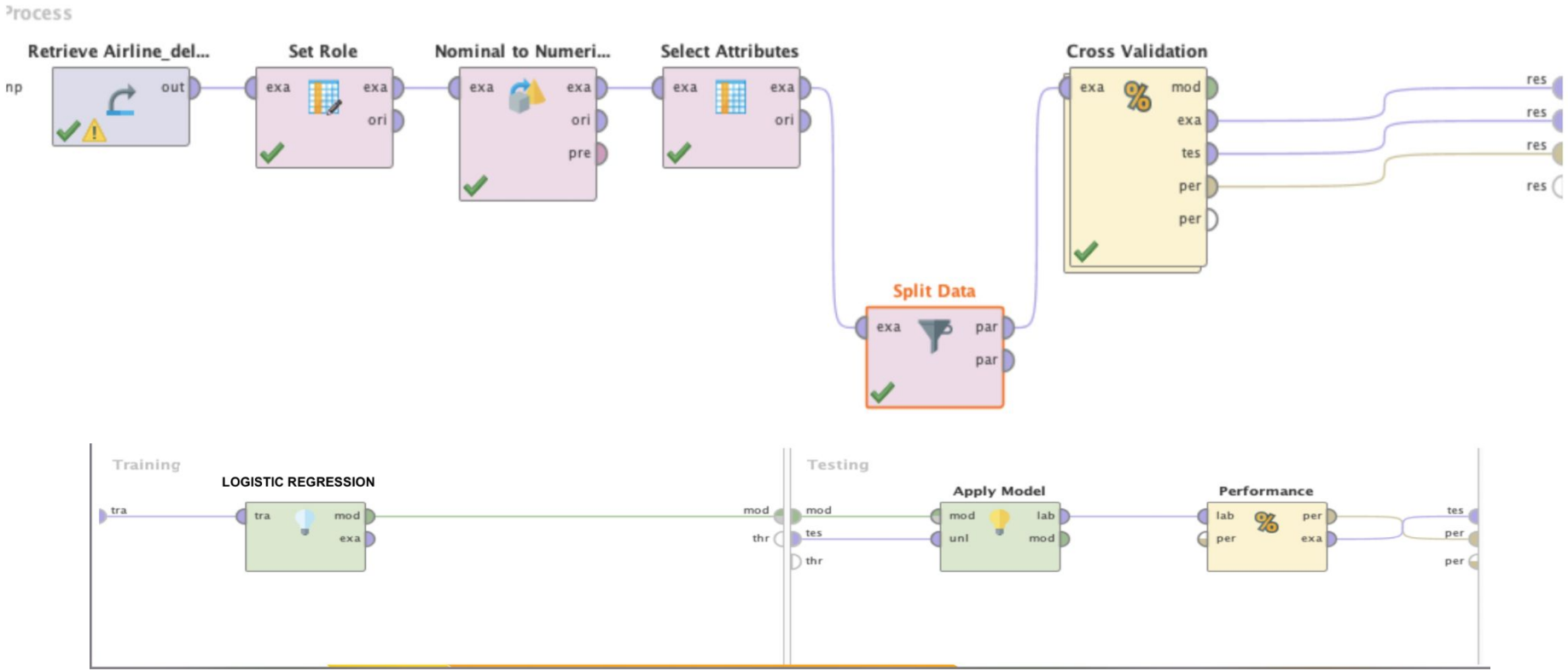
Row No.	combinedy...	prediction(...	confidence(N)	confidence(Y)	cost	combinedyears.uniquecarrier	combinedyears.year
1	N	Y	0.476	0.524	0.047	NW	2008
2	N	Y	0.483	0.517	0.034	WN	2008
3	Y	Y	0.478	0.522	0.044	MQ	2008
4	Y	Y	0.483	0.517	0.034	AA	2008
5	N	N	0.625	0.375	0.250	AS	2008
6	Y	N	0.564	0.436	0.129	EV	2008
7	N	Y	0.483	0.517	0.034	AA	2008
8	N	Y	0.483	0.517	0.034	WN	2008
9	Y	Y	0.497	0.503	0.006	9E	2008
10	N	Y	0.483	0.517	0.034	AA	2008
11	Y	Y	0.483	0.517	0.034	WN	2008
12	Y	Y	0.483	0.517	0.034	WN	2008
13	N	N	0.581	0.419	0.161	FL	2008
14	Y	Y	0.466	0.534	0.068	CO	2008
15	Y	N	0.527	0.473	0.053	DL	2008
16	N	N	0.581	0.419	0.161	FL	2008
17	N	N	0.527	0.473	0.053	DL	2008
18	Y	Y	0.478	0.522	0.044	MQ	2008

LOGISTIC REGRESSION!

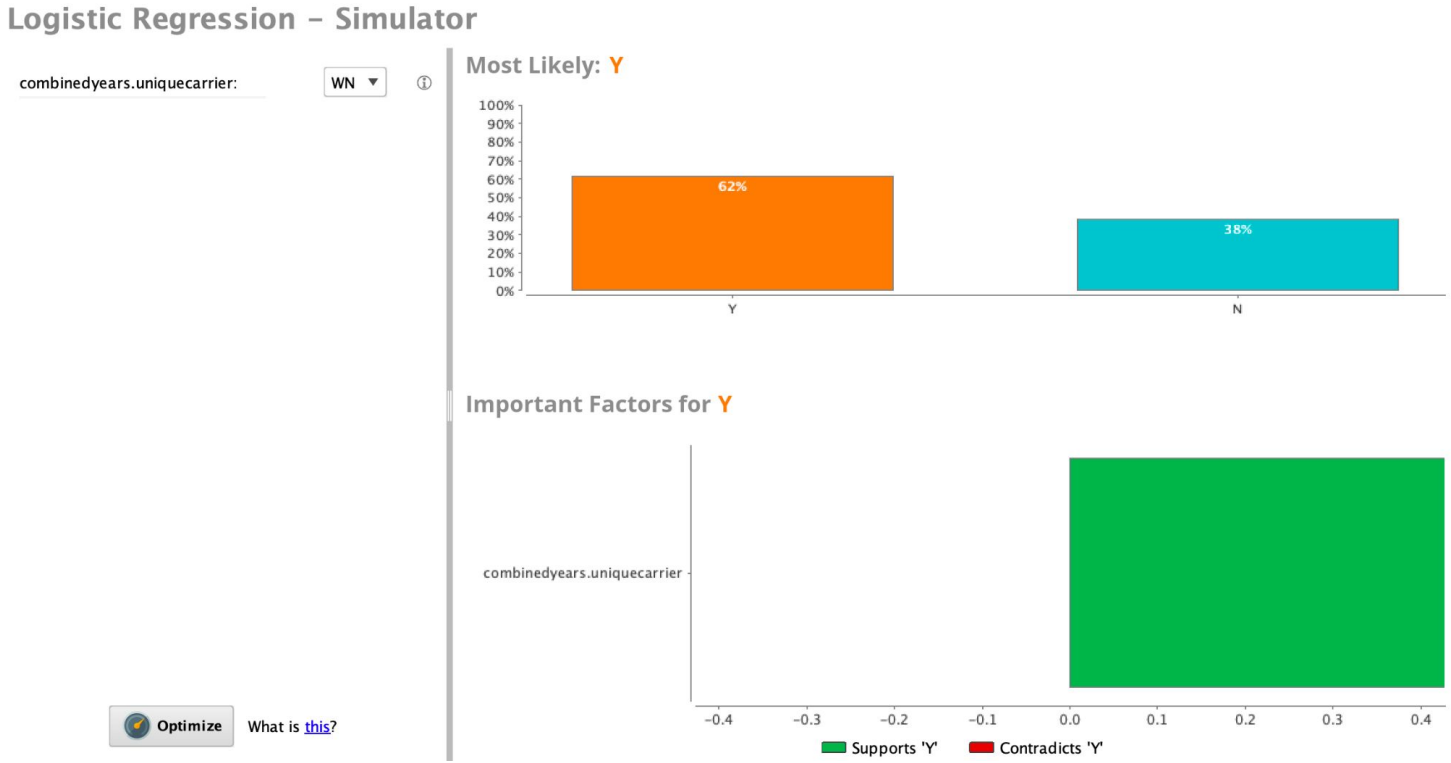
What is Logistic Regression?

- Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.
- Applications. Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression.

PROCESS


Process
Retrieve Airline_del...
out
Set Role
exa
ori
Nominal to Numeri...
pre
Select Attributes
Cross Validation
mod
tes
per
res
Split Data
par
Training
LOGISTIC REGRESSION
tra
Testing
Apply Model
lab
unl
Performance

Model Simulation



PERFORMANCE

- ❑Classification error: 41.3%
- ❑Accuracy(Cross-Validation): 58.6%
  - ❑ Predicted Y (58.52%)
    - 3921 Predicted true Y
    - 2779 Predicted true N
  - ❑ Predicted N (64.94%)
    - 27 Predicted true Y
    - 50 Predicted true N

Logistic Regression – Performance

Profits

Profits from Model: 1,165 Profits for Best Option (Y): 1,119 Gain: 46 Show Costs / Benefits...

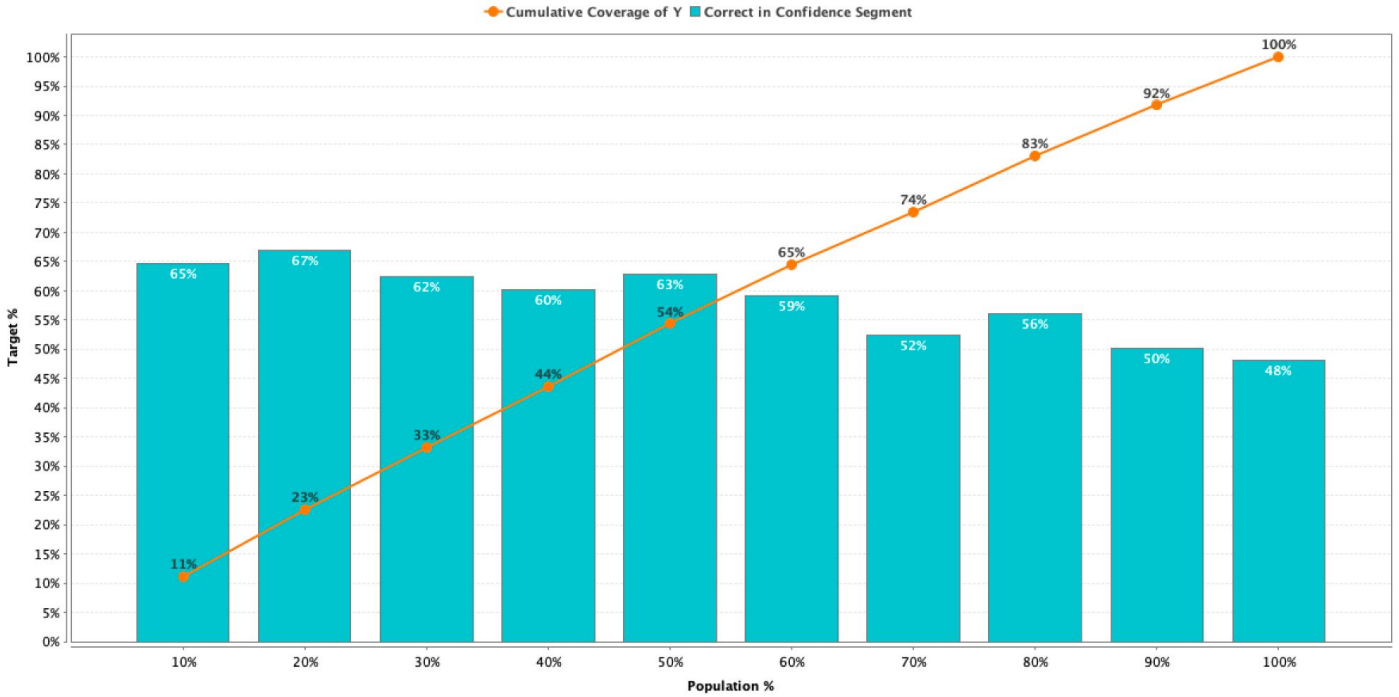
Performances

Criterion	Value	Standard Deviation
Accuracy	58.6%	± 1.2%
Classification Error	41.4%	± 1.2%
AUC	56.5%	± 0.6%
Precision	58.5%	± 1.1%
Recall	99.3%	± 0.4%
F Measure	73.6%	± 1.0%
Sensitivity	99.3%	± 0.4%
Specificity	1.8%	± 0.6%

Confusion Matrix

	true N	true Y	class precision
pred. N	50	27	64.94%
pred. Y	2779	3921	58.52%
class recall	1.77%	99.32%	

LIFT CHART

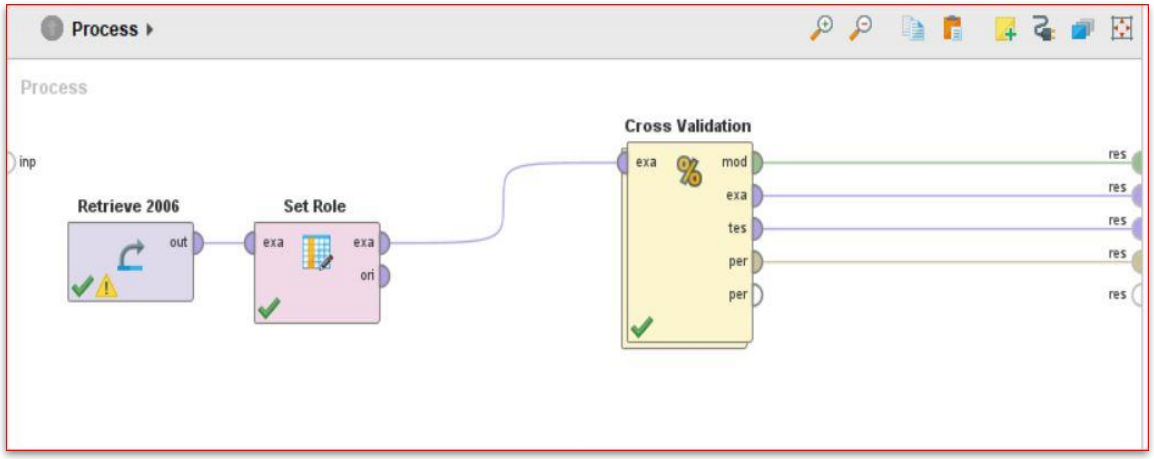

Logistic Regression - Lift Chart
Cumulative Coverage of Y
Correct in Confidence Segment
Target %
Population %
100%
95%
90%
85%
80%
75%
70%
65%
60%
55%
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
65%
67%
62%
60%
63%
59%
52%
56%
50%
48%
11%
23%
33%
44%
54%
65%
74%
83%
92%
100%

PREDICTION

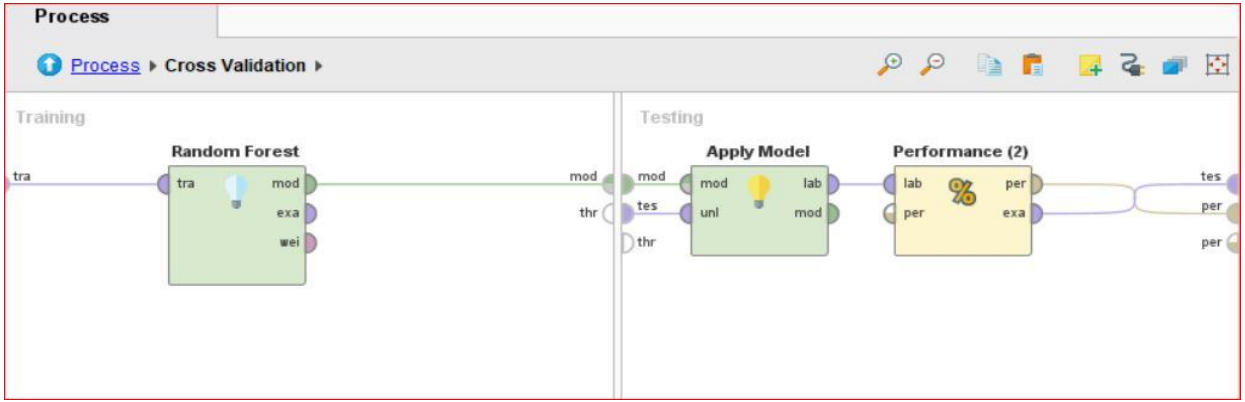
Logistic Regression – Predictions

Row No.	combinedy...	prediction(...	confidence(...	confidence(...	cost	combinedy...
1	Y	Y	0.376	0.624	0.248	AA
2	N	Y	0.439	0.561	0.122	OO
3	Y	Y	0.371	0.629	0.258	CO
4	Y	Y	0.400	0.600	0.199	MQ
5	N	Y	0.388	0.612	0.223	UA
6	Y	Y	0.400	0.600	0.199	MQ
7	N	Y	0.415	0.585	0.169	DL
8	N	Y	0.415	0.585	0.169	DL
9	Y	N	0.513	0.487	0.027	AQ
10	Y	Y	0.425	0.575	0.150	EV
11	N	Y	0.376	0.624	0.248	AA
12	Y	Y	0.439	0.561	0.122	OO
13	Y	Y	0.383	0.617	0.234	WN
14	N	Y	0.376	0.624	0.248	AA
15	N	Y	0.401	0.599	0.198	NW
16	Y	Y	0.383	0.617	0.234	WN
17	N	Y	0.439	0.561	0.122	OO
18	Y	Y	0.371	0.629	0.258	CO

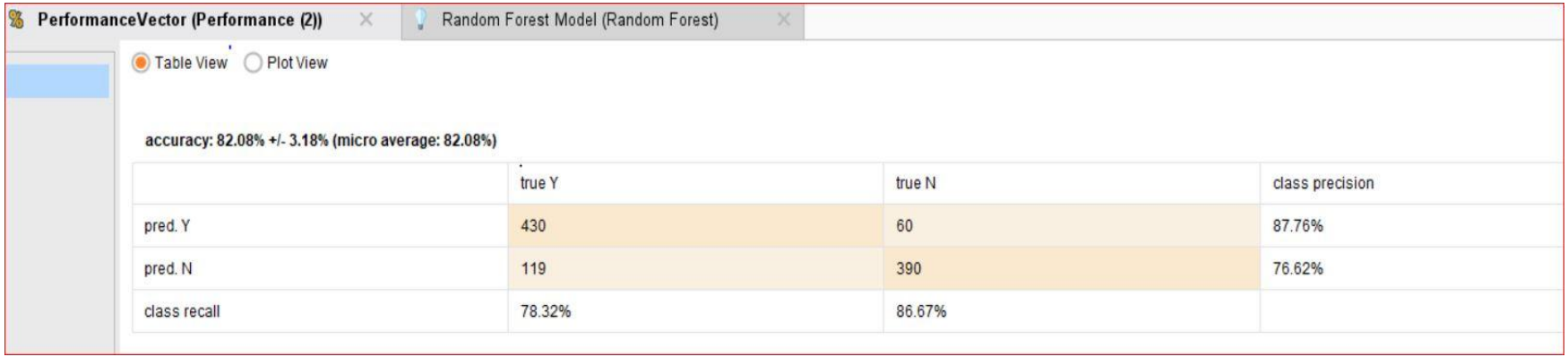
RANDOM FOREST!


Process
Process
inp
Retrieve 2006
out
Set Role
exa
exa
ori
Cross Validation
exa
mod
exa
tes
per
per
res
res
res
res
res

DESIGN


Process
Process
Cross Validation
Training
Testing
Random Forest
tra
tra
mod
exa
wei
mod
thr
Apply Model
mod
tes
thr
mod
unl
lab
mod
Performance (2)
lab
per
per
exa
tes
per
per

TRAINING MODEL - RANDOM FOREST



PerformanceVector (Performance (2)) | Random Forest Model (Random Forest)

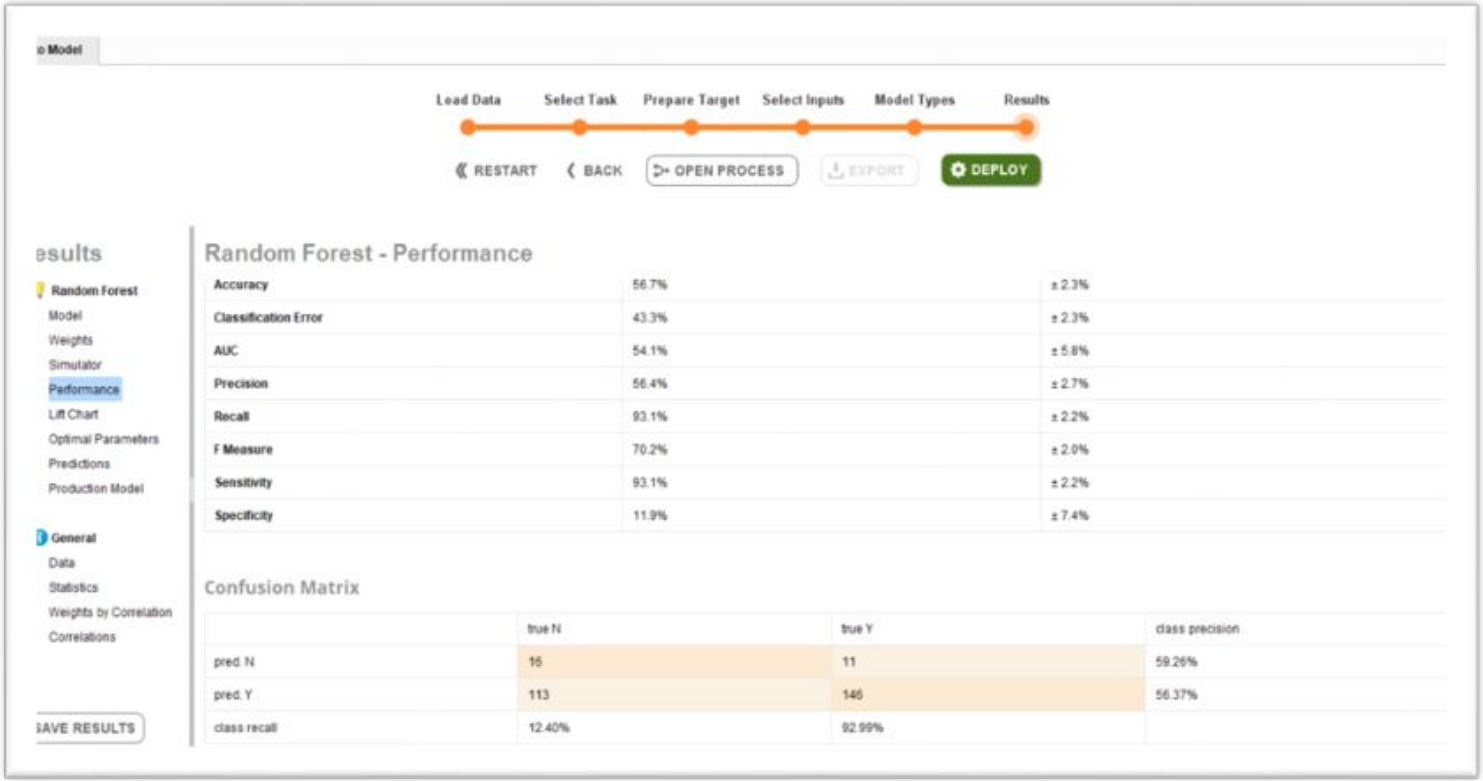
Table View | Plot View

accuracy: 82.08% +/- 3.18% (micro average: 82.08%)

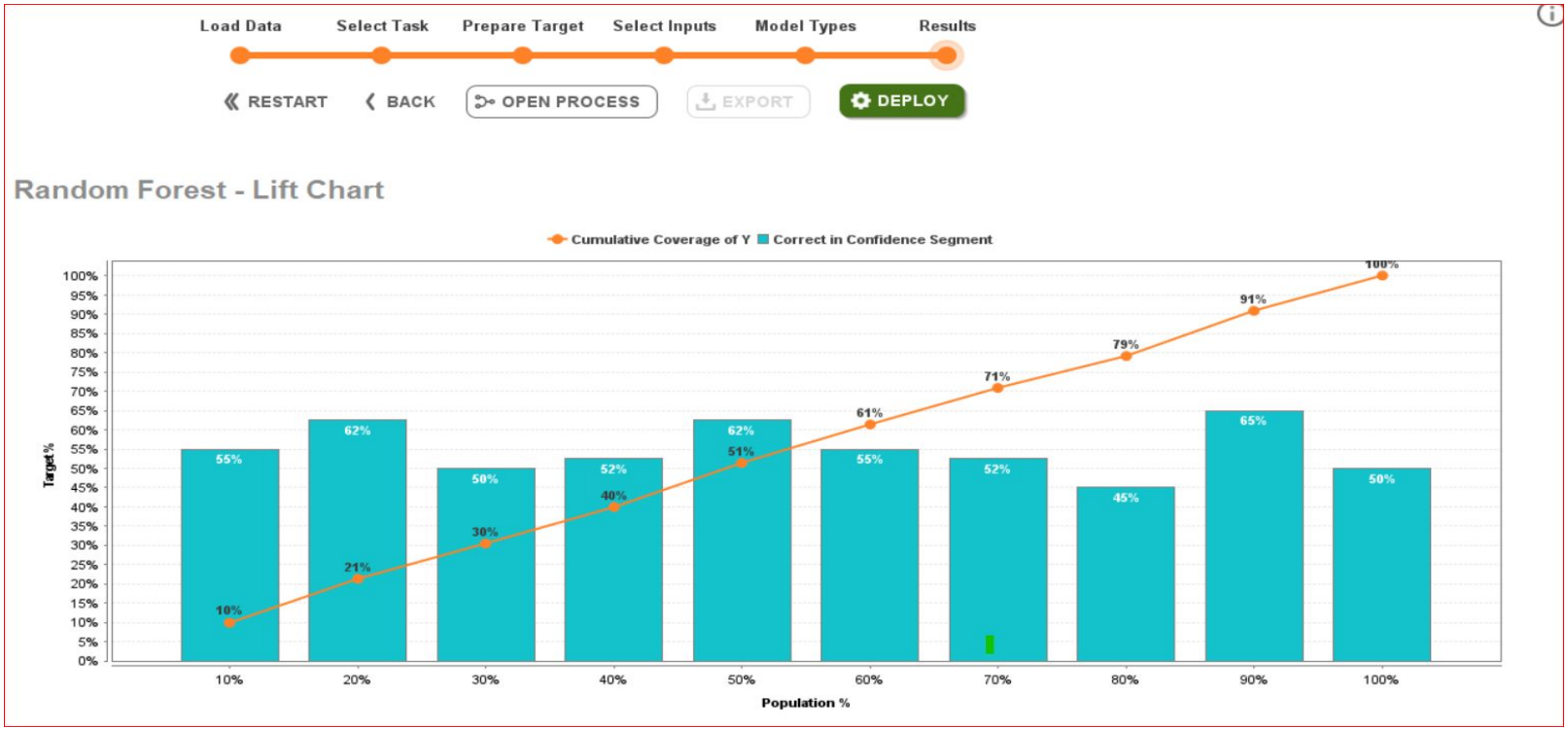
	true Y	true N	class precision
pred. Y	430	60	87.76%
pred. N	119	390	76.62%
class recall	78.32%	86.67%	

- Classification error: 43.3%
- Accuracy(Cross-Validation): 82.08%
  - Predicted Y (87.76%)
    - 430 Predicted true Y
    - 60 Predicted true N
  - Predicted N (76.72%)
    - 119 Predicted true Y
    - 390 Predicted true N

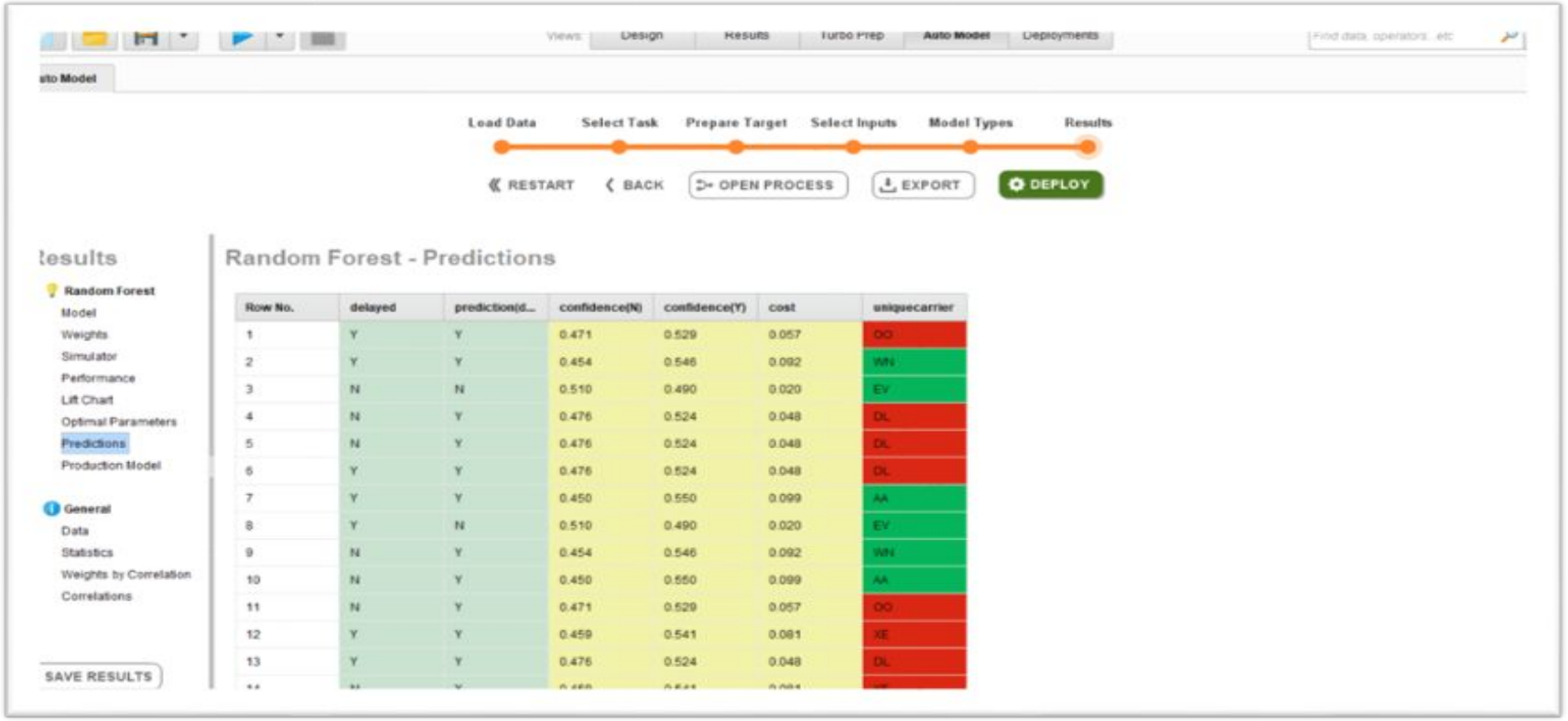
RANDOM FOREST - PERFORMANCE



RANDOM FOREST - LIFT CHART



RANDOM FOREST - PREDICTIONS



Row No.	delayed	prediction(d...	confidence(N)	confidence(Y)	cost	uniquecarrier
1	Y	Y	0.471	0.529	0.057	OO
2	Y	Y	0.454	0.546	0.092	WN
3	N	N	0.510	0.490	0.020	EV
4	N	Y	0.476	0.524	0.048	DL
5	N	Y	0.476	0.524	0.048	DL
6	Y	Y	0.476	0.524	0.048	DL
7	Y	Y	0.450	0.550	0.099	AA
8	Y	N	0.510	0.490	0.020	EV
9	N	Y	0.454	0.546	0.092	WN
10	N	Y	0.450	0.550	0.099	AA
11	N	Y	0.471	0.529	0.057	OO
12	Y	Y	0.459	0.541	0.081	XE
13	Y	Y	0.476	0.524	0.048	DL

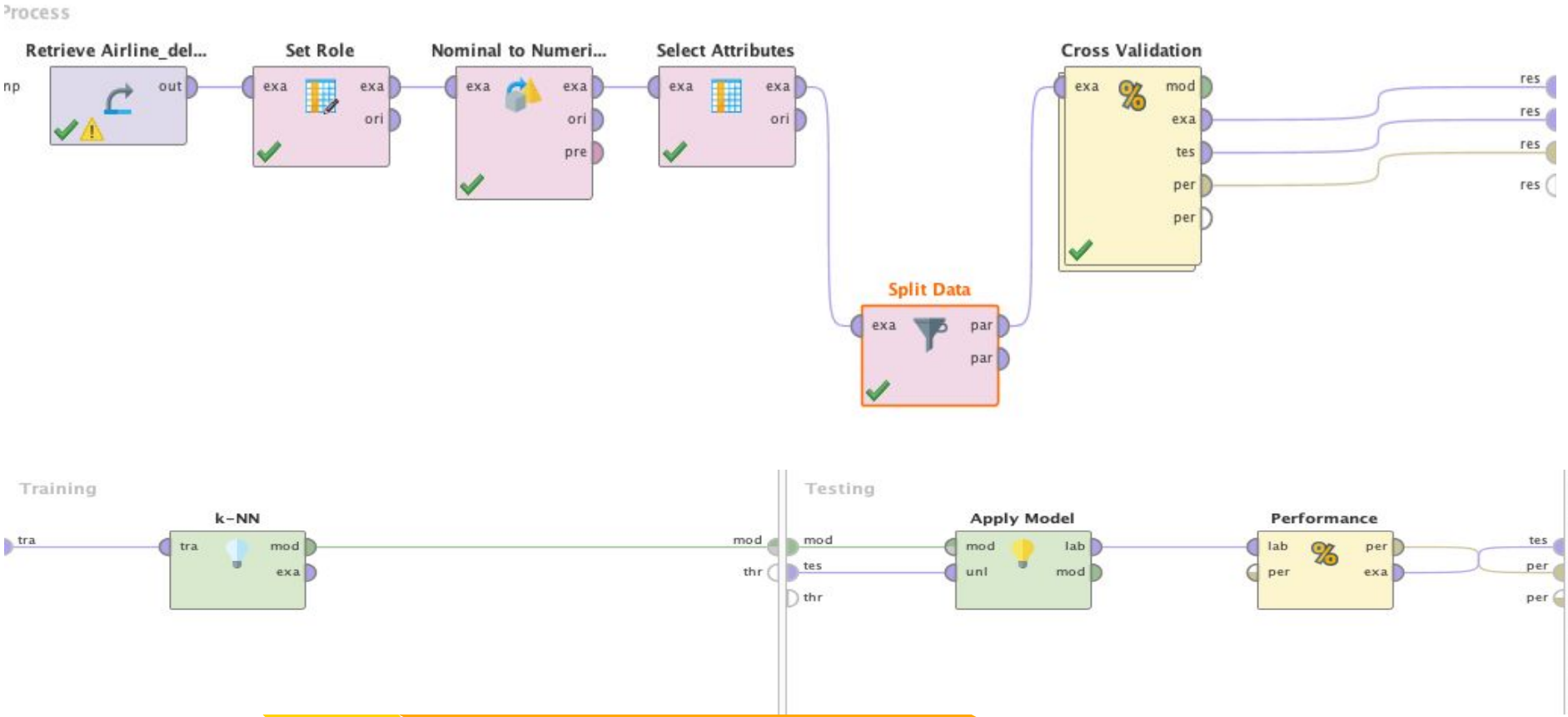
K-NEAREST NEIGHBOUR!

WHAT IS KNN?

- The KNN algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problem

- Why is KNN a Lazy Algorithm?

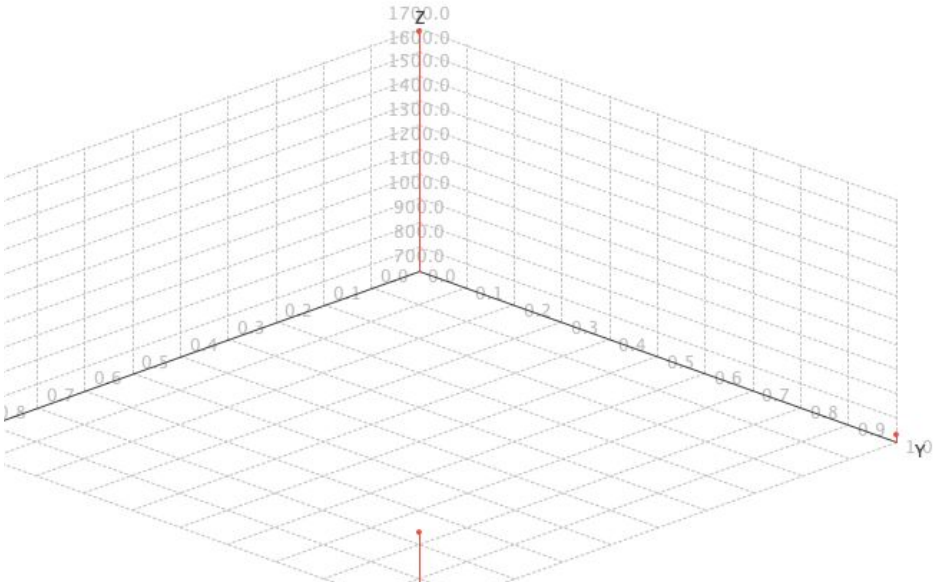
DESIGN


Process
Retrieve Airline_del...
Set Role
Nominal to Numeri...
Select Attributes
Cross Validation
Split Data
Training
k-NN
Testing
Apply Model
Performance

PERFORMANCE

accuracy: 64.52%

	true Y	true N
pred. Y	1684	761
pred. N	729	1026
class recall	69.79%	57.41

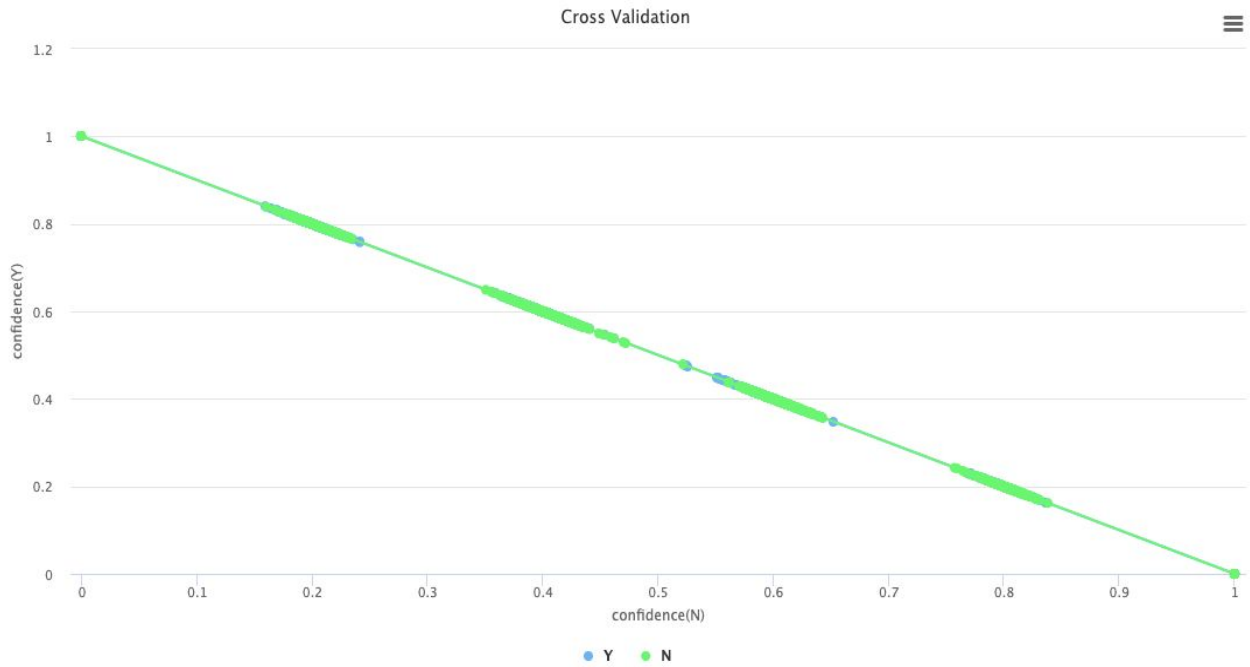


- The Accuracy of the model is 64.52%
- With Class Recall values shown as above
- The performance of the model is good in general

PREDICTION

Row No.	Delays	prediction(...	confidence(...	confidence(...	DepTime	CRSDepTime	ArrTime	CRSArrTime	ActualElap...	CRSElapse...	AirTime	ArrDelay
1	Y	N	0.396	0.604	730	730	1124	1119	174	169	0	5
2	N	N	0.192	0.808	830	830	938	942	68	72	0	-4
3	N	N	0.406	0.594	740	740	856	905	136	145	0	-9
4	Y	N	0.388	0.612	1318	1310	1417	1415	59	65	0	2
5	N	N	0.395	0.605	1300	1300	1411	1421	71	81	0	-10
6	Y	N	0.181	0.819	707	700	744	747	97	107	0	-3
7	Y	Y	0.802	0.198	1901	1814	2056	2014	115	120	0	42
8	Y	N	0.400	0.600	1526	1525	1629	1632	63	67	0	-3
9	N	N	0.426	0.574	656	700	746	755	50	55	0	-9
10	Y	Y	0.600	0.400	2318	2315	620	630	242	255	0	-10
11	N	N	0.364	0.636	615	615	710	710	55	55	0	0
12	N	Y	0.602	0.398	1911	1915	2024	2032	73	77	0	-8
13	N	Y	0.598	0.402	2043	2045	2333	2338	110	113	0	-5
14	Y	N	0.193	0.807	700	700	1313	1301	253	241	0	12
15	N	Y	0.820	0.180	1734	1735	1924	1928	110	113	0	-4

PREDICTION CONFIDENCE INTERVAL



- The Range of confidence interval values of N tends to be on a larger scale when compared to Y values
- Scatter Plot & Bubble plot is used to display such outcome
- The algorithm is a good predicting model in general

SUPPORT VECTOR MACHINE !

DATA WRANGLING

```
#Adding the Delay column into the combined table
Airline_delay.loc[(Airline_delay['ArrDelay']>0.0) | (Airline_delay['DepDelay']>=0.0),'Delays'] = 'Y'
Airline_delay.loc[(Airline_delay['ArrDelay']<=0.0) & (Airline_delay['DepDelay']<=0.0),'Delays'] = 'N'

#Missing values in the dataset
Airline_delay.isna().sum()

#Replacing the missing values with Zero
Airline_delay.fillna(value=0,inplace=True)
```

Adding Delay Column

Replace the missing value with 0

```
Index(['Year', 'Month', 'DayofMonth', 'DayOfWeek', 'DepTime', 'CRSDepTime',
       'ArrTime', 'CRSArrTime', 'UniqueCarrier', 'FlightNum', 'TailNum',
       'ActualElapsedTime', 'CRSElapsedTime', 'AirTime', 'ArrDelay',
       'DepDelay', 'Origin', 'Dest', 'Distance', 'TaxiIn', 'TaxiOut',
       'Cancelled', 'CancellationCode', 'Diverted', 'CarrierDelay',
       'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay',
       'Delays'],
      dtype='object')
```

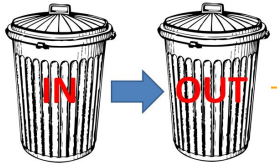
Columns after adding Delays attribute

```
Year                    0
Month                   0
DayofMonth              0
DayOfWeek               0
DepTime               127
CRSDepTime              0
ArrTime               141
CRSArrTime              0
UniqueCarrier           0
FlightNum               0
TailNum              2000
ActualElapsedTime     141
CRSElapsedTime          0
AirTime              2116
ArrDelay              141
DepDelay              127
Origin                  0
Dest                    0
Distance                4
TaxiIn               2000
TaxiOut              2000
Cancelled               0
CancellationCode     5927
Diverted                0
CarrierDelay         3000
WeatherDelay         3000
NASDelay             3000
SecurityDelay        3000
LateAircraftDelay    3000
Delays                131
dtype: int64
```

```
Year                 0
Month                0
DayofMonth           0
DayOfWeek            0
DepTime              0
CRSDepTime           0
ArrTime              0
CRSArrTime           0
UniqueCarrier        0
FlightNum            0
TailNum              0
ActualElapsedTime    0
CRSElapsedTime       0
AirTime              0
ArrDelay             0
DepDelay             0
Origin               0
Dest                 0
Distance             0
TaxiIn               0
TaxiOut              0
Cancelled            0
CancellationCode     0
Diverted             0
CarrierDelay         0
WeatherDelay         0
NASDelay             0
SecurityDelay        0
LateAircraftDelay    0
Delays               0
dtype: int64
```

Before & After Missing Values

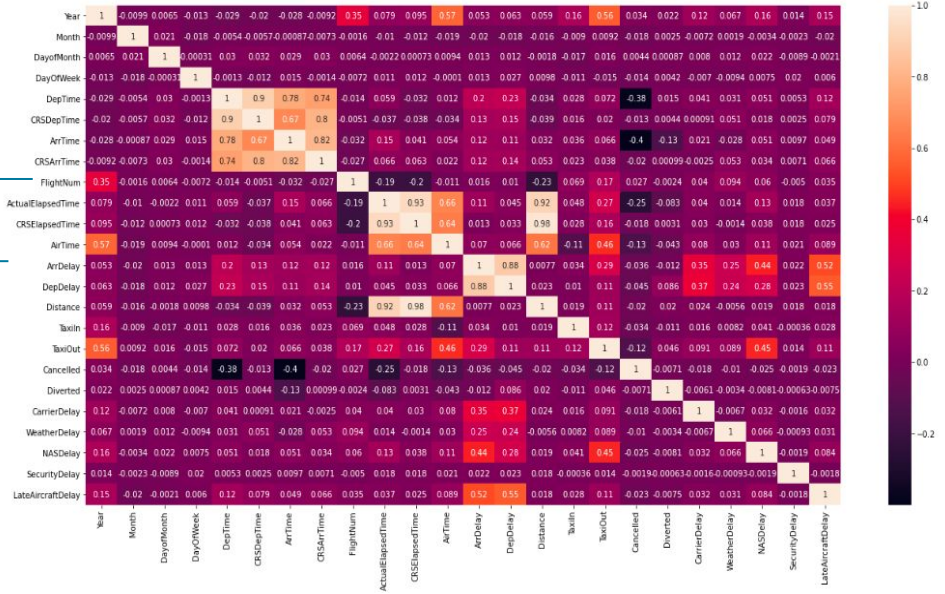
FEATURE ENGINEERING



GIGO

Heatmap was used for feature engineering for selection of independent variables

There were 8 columns that were selected using heatmap for model training



	DepTime	CRSDepTime	ArrTime	CRSArrTime	ActualElapsedTime	CRSElapsedTime	AirTime	Delays
4209789	730.0	730	1124.0	1119	174.0	169.0	0.0	Y
2377031	1430.0	1430	1523.0	1531	53.0	61.0	0.0	N
120245	2043.0	2045	2319.0	2339	156.0	174.0	0.0	N
4854736	1224.0	1225	1329.0	1333	125.0	128.0	0.0	N
4115444	1644.0	1645	1902.0	1836	198.0	171.0	0.0	Y
...	...	...	...	...	...	...	...	...
234588	1014.0	1013	1319.0	1323	125.0	130.0	107.0	Y
1675188	1815.0	1810	2221.0	2140	186.0	150.0	143.0	Y
2007786	2008.0	2000	2234.0	2220	266.0	260.0	241.0	Y
2176206	1456.0	1500	1621.0	1620	85.0	80.0	60.0	Y
1158142	2303.0	2310	648.0	728	285.0	318.0	264.0	N

The DataFrame is shown for evidence

MODEL

```
#Training & Testing the data for model deployement
x=Airline_delay2.drop("Delays",axis=1)
y=Airline_delay2['Delays']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,random_state=100)

#Linear SVC
from sklearn.svm import LinearSVC
clf = svm.LinearSVC()
clf.fit(x_train, y_train)
clf.predict(x_test)
clf.score(x_test,y_test)

#Cross-Validation
from sklearn.model_selection import KFold
kf = KFold(n_splits=10)
for train_index ,test_index in kf.split([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18]):
    print(train_index ,test_index)

def get_score(model,x_train,y_train,x_test,y_test):
model.fit(x_train,y_train)
return model.score(x_test,y_test)
get_score(clf,x_train,y_train,x_test,y_test)
```

```
get_score(clf,x_train,y_train,x_test,y_test)*100
```

```
/Users/zee/opt/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:976: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
57.41898806139852
```

Splitting the data into 70:30 Ratio & Using Cross-Validation for accurate results.

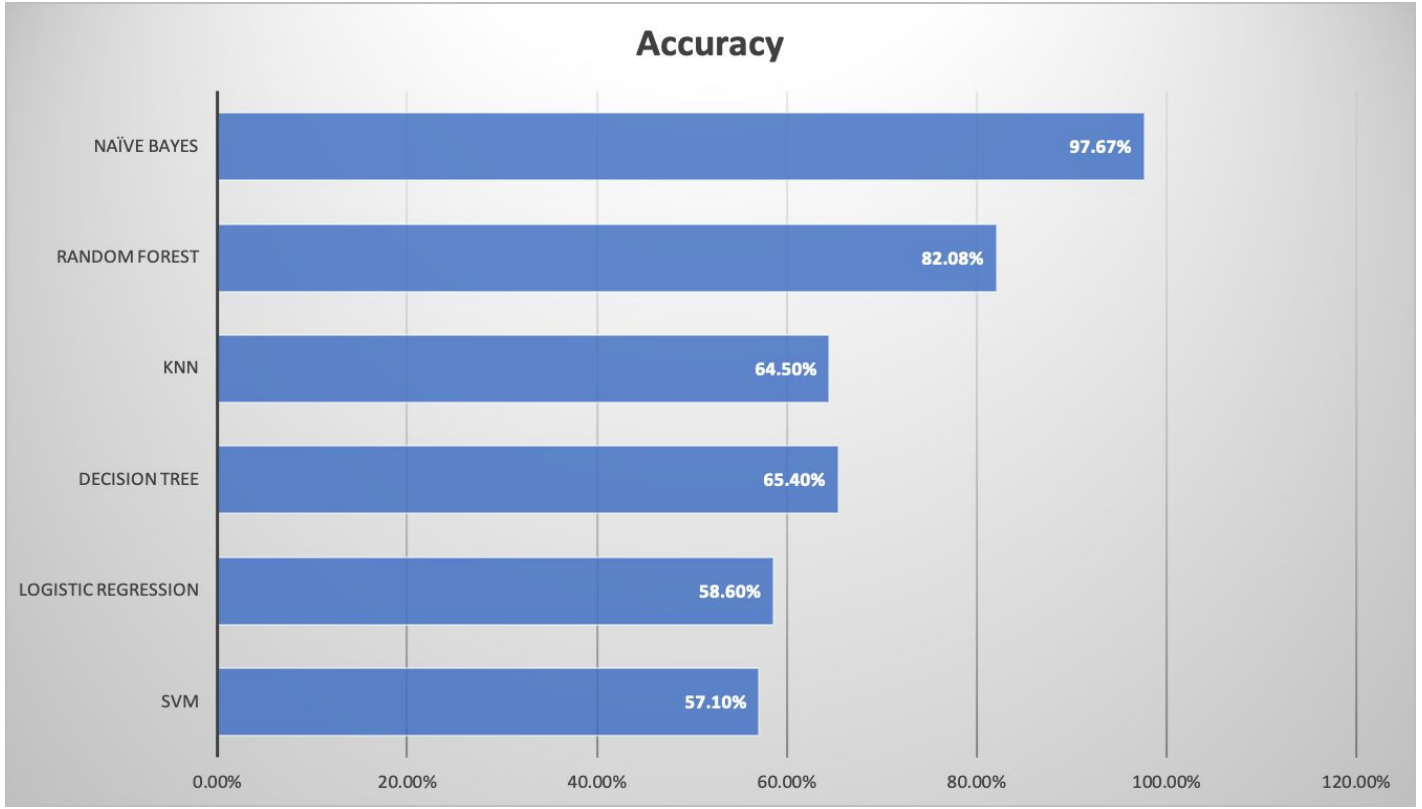
Determining the score model for the given SVM (Linear) using sklearn

Used Rapidminer To for checking the accuracy & Performance of the model

accuracy: 66.67%

	true Y	true N	class precision
pred. Y	1766	769	69.66%
pred. N	599	970	61.82%
class recall	74.67%	55.78%	

ACCURACY COMPARISON



FINDINGS ON NEW DATA

Row No.	prediction(...	confidence(...	confidence(...	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarr...
1	N	0.446	0.554	2023	1	17	6	?	630	?	915	UA
2	Y	0.838	0.162	2023	2	28	6	?	1820	?	2202	DL
3	Y	0.609	0.391	2023	3	16	2	?	1241	?	1338	DL
4	Y	0.686	0.314	2023	4	7	3	?	1305	?	1544	CO
5	Y	0.509	0.491	2023	5	13	4	?	1014	?	958	MQ
6	Y	0.621	0.379	2023	6	19	6	?	730	?	1240	DL
7	Y	0.700	0.300	2023	7	6	2	?	1450	?	1615	UA
8	Y	0.727	0.273	2023	8	31	2	?	805	?	1013	UA
9	Y	0.775	0.225	2023	9	28	2	?	1610	?	1942	TZ
10	Y	0.551	0.449	2023	10	12	2	?	959	?	1117	TZ
11	Y	0.787	0.213	2023	11	10	3	?	1740	?	1924	EV
12	Y	0.505	0.495	2023	12	16	4	?	813	?	1008	DL

Conclusion

After careful training and observing testing results from 6 classification models, we have come to make the conclusion that

- NB would be a best generalized model according to the prediction results on our unseen data with considerable accuracy rate.
- The independent variables were - DepTime, CRSDepTime, AirTime, CRSArrTime, CRSElapsedTime, ActualElapsedTime,ArrDelay

Questions?