Model Training & Testing

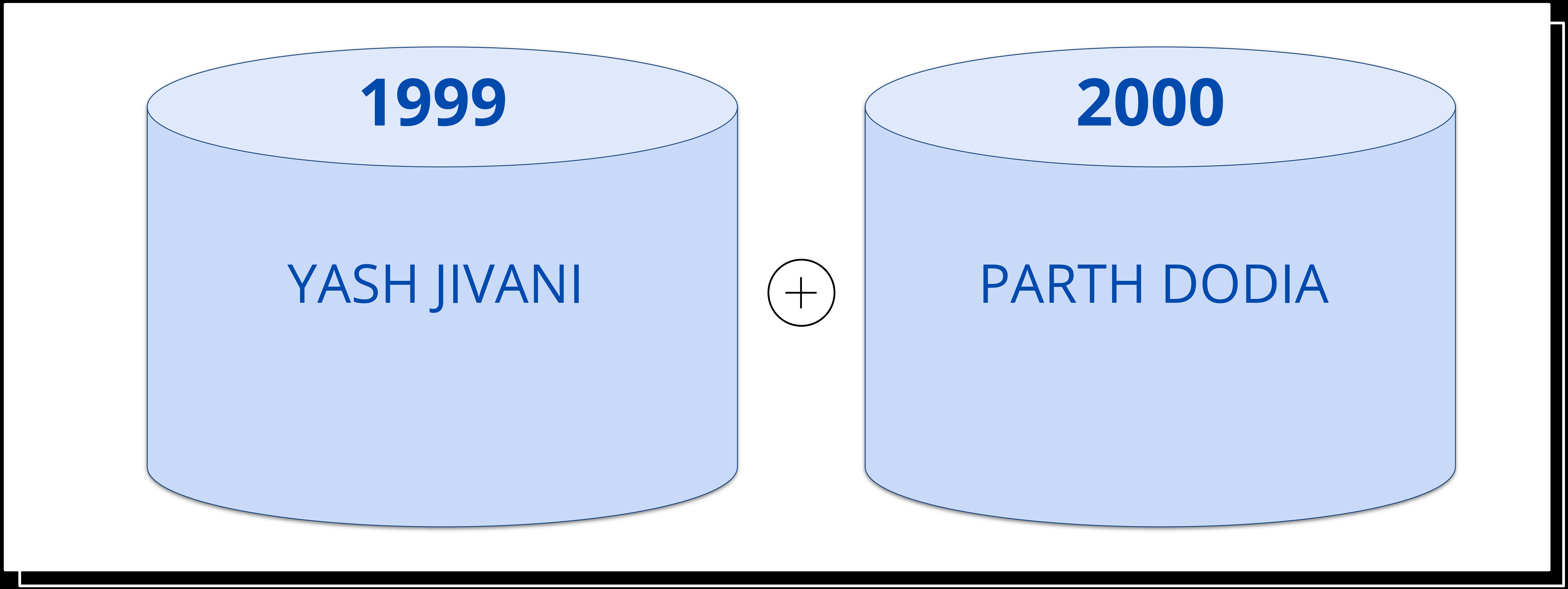
Big Data Analytics Project Phase 2

YASH JIVANI
PARTH DODIA

DATA Modelling TASKS

- Get a random sample of 1,000 records from each year.
- Add a new column to the table and label it Delay
- Populate the Delay column as follows:
    a. if the record has a value of zero or less in the ArrDelay and DepDelay columns, put a “N” in the Delay column
    b. otherwise, put a “Y” in the Delay column
- Combining the file with the other sample year files from each group member.
- Choosing a different model that can predict whether new unknown records will have any delays or not at all
- Training, validating and testing the analytics model with different segments of the combined sample data.
- Agreeing on the best team model to predict the delay of twelve flight records provided by the instructor.

Records



DATA SOURCE

Data Expo 2009: Airline on Time Data

The data represents flight arrival and departure details for all commercial flights within the USA for year 1999 and 2000

Download Link

Data Expo 2009: Harward - Airline on Time Data

Data Processing

Reading the csv files and cleaning the data.

```
import pandas as pd
```

```
#reading the data from the file of year 1999
df_1999_all = pd.read_csv('1999.csv.bz2')
```

```
#reading the data from the file of year 2000
df_2000_all = pd.read_csv('2000.csv.bz2')
```

```
#not to add the record which has empty values of arrival and departure Delay
df_1999_all = df_1999_all[~df_1999_all.ArrDelay.isnull() & ~df_1999_all.DepDelay.isnull()]
df_2000_all = df_2000_all[~df_2000_all.ArrDelay.isnull() & ~df_2000_all.DepDelay.isnull()]
```

```
df_1999_all.shape
```

```
(5360018, 29)
```

```
df_2000_all.shape
```

```
(5481303, 29)
```

Data Processing

Getting 1000 sample records from each file and combining the data

```
# take sample 1000 records
df_1999_sample = df_1999_all.sample(1000)
df_2000_sample = df_2000_all.sample(1000)
```

```
df_sample = pd.concat([df_1999_sample,df_2000_sample])
```

```
df_sample.reset_index(inplace=True)
```

```
df_sample.drop(columns=['index'], axis=1,inplace=True)
```

```
#adding the Delay column and adding the values in it using require condition
df_sample['Delay'] = df_sample.apply(lambda x: 'N' if x.ArrDelay <= 0 and x.DepDelay <= 0 else 'Y', axis=1)
```

```
#getting the combined data into new csv file
df_sample.to_csv('sample_1999+2000.csv', index=False)
```

Data Processing

```
df = pd.read_csv('sample_1999+2000.csv')
```

```
df
```

	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay
0	1999	9	8	3	1835.0	1823	2050.0	2048	AS	281	N937AS	135.0	145.0	117.0	2.0
1	1999	4	12	1	1632.0	1635	1844.0	1835	DL	76	N173DN	132.0	120.0	96.0	9.0
2	1999	12	14	2	829.0	830	1108.0	1104	DL	847	N906DL	159.0	154.0	123.0	4.0
3	1999	3	18	4	750.0	750	949.0	950	WN	1767	N505	59.0	60.0	49.0	-1.0
4	1999	8	28	6	808.0	810	902.0	910	DL	1156	N675DL	54.0	60.0	38.0	-8.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	2000	5	19	5	1245.0	1225	1433.0	1435	NW	345	N515US	288.0	310.0	269.0	-2.0
1996	2000	5	9	2	1427.0	1425	1642.0	1634	UA	918	N529UA	135.0	129.0	111.0	8.0
1997	2000	10	9	1	918.0	920	1042.0	1040	NW	707	N986US	144.0	140.0	114.0	2.0
1998	2000	11	8	3	1238.0	1233	1345.0	1350	AA	1057	NR33AA	67.0	77.0	42.0	-5.0
1999	2000	12	23	6	1443.0	1250	1652.0	1440	DL	2063	N911DL	129.0	110.0	89.0	132.0

2000 rows × 30 columns

Combined Data

Decision Tree

What is Decision Tree?

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

Why Decision tree?

Decision trees are widely used for data analytics and machine learning because they can break down complex data into more manageable parts. They're often used in these fields for prediction analysis, data classification, and regression.

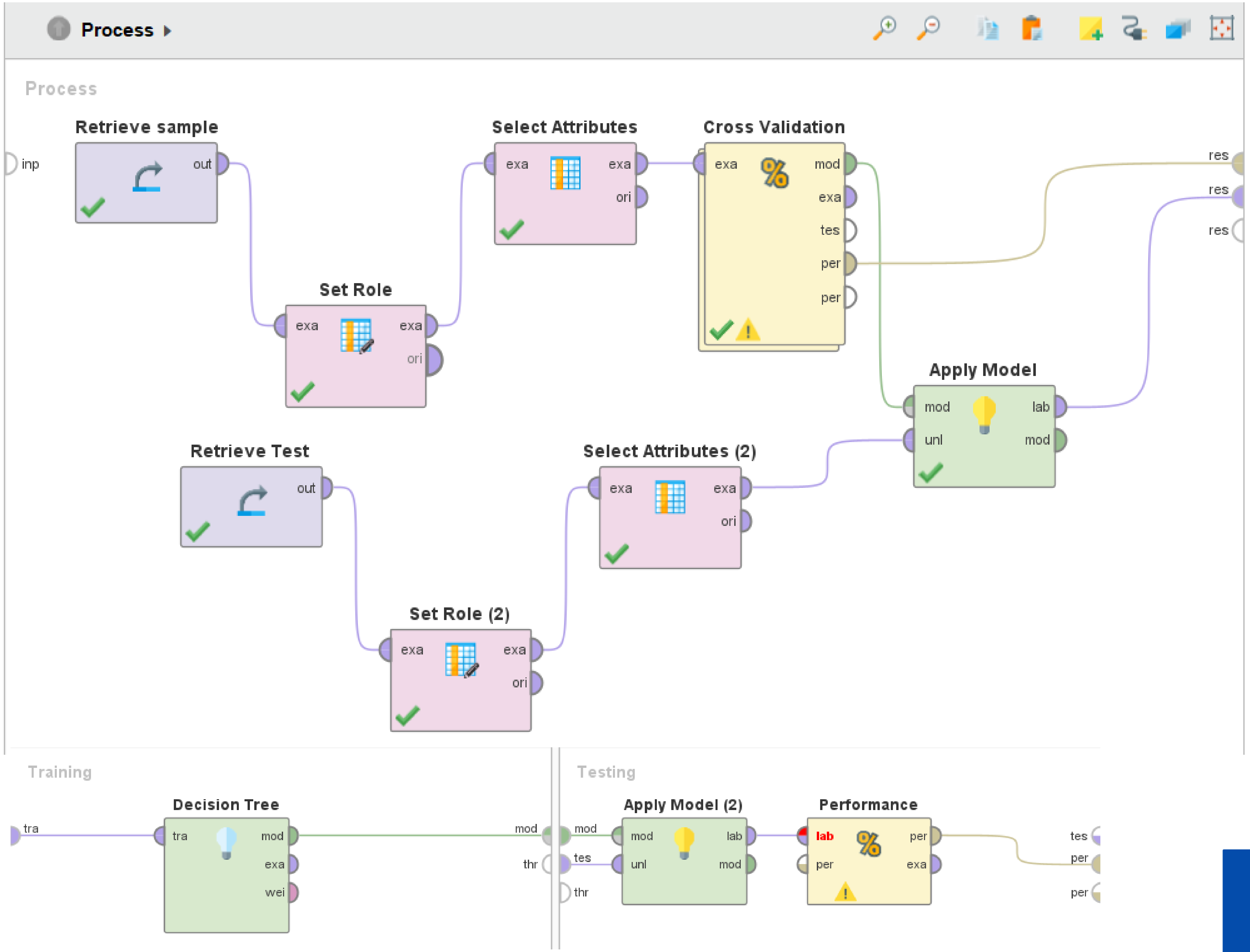
Testing Data

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance
2		1	30	6		1920	2016	2030	US	1541	N275AU		70	42	-14		ROC	PHL	257
3		2	6	6		728	957	944	TW	606	N921L		76	53	13		STL	TYS	405
4		3	17	3		0	1330	0	UA	1660	N951UA		205	182	-10		DEN	PHL	1557
5		4	24	6		700	920	940	UA	462	N998UA		100	74	-20		ORD	CLT	599
6		5	1	7		1205	1306	1312	TW	447	NA		127	NA	-6		ORF	STL	784
7		6	7	2		945	1044	1048	US	253	NA		63	NA	-4		PIT	LEX	289
8		7	14	4		1455	1730	1714	CO	1098	NA		139	NA	16		IAH	ORD	925
9		8	22	1		1750	1938	1942	DL	922	NA		112	NA	-4		CVG	LGA	585
10		9	12	3		1500	1602	1605	OO	3691	N565SW		65	52	-3		SUN	SLC	223
11		10	14	7		1500	1715	1710	MQ	4471	N820AE		130	109	5		ORD	PNS	794
12		11	6	2		730	1022	1013	NW	1674	N613NW		163	147	9		AUS	MSP	1042
13		12	14	5		2030	2130	2144	DL	1969	N916DE		74	40	-14		LGA	DCA	214
14																			

T	U	V	W	X	Y	Z	AA	AB	AC	AD
TaxiIn	TaxiOut	Cancelled	CancellationCode	Diverted	CarrierDelay	WeatherDelay	NASDelay	SecurityDelay	LateAircraftDelay	Delayed (Y or N)
8	13									
4	19									
3	13									
2	8									
NA	NA									
NA	NA									
NA	NA									
NA	NA									
4	8									
3	18									
11	20									
4	17									

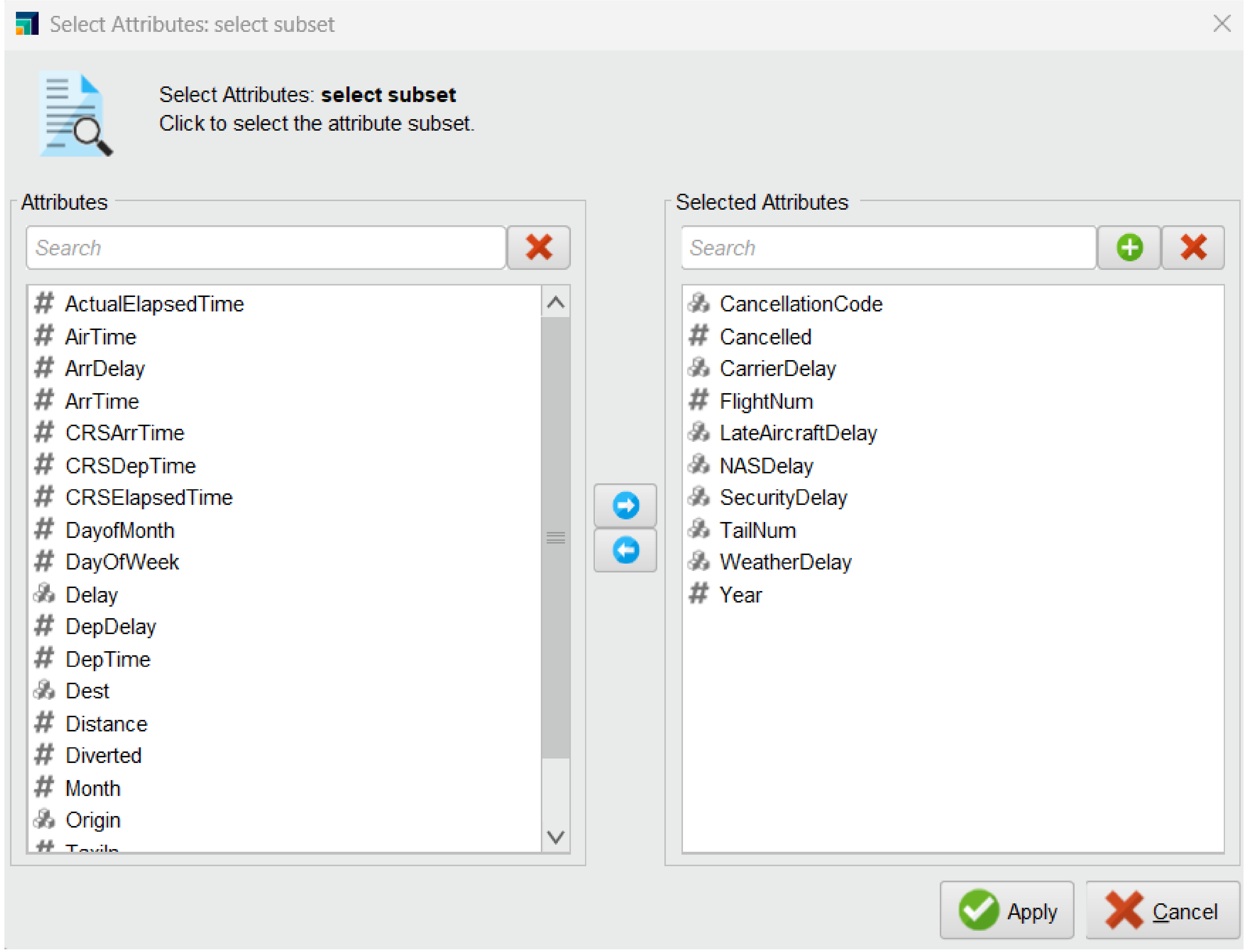
Training and Testing

Training the model in Rapid Miner using the combined dataset.



Training and Testing

Selecting the appropriate attributes for training



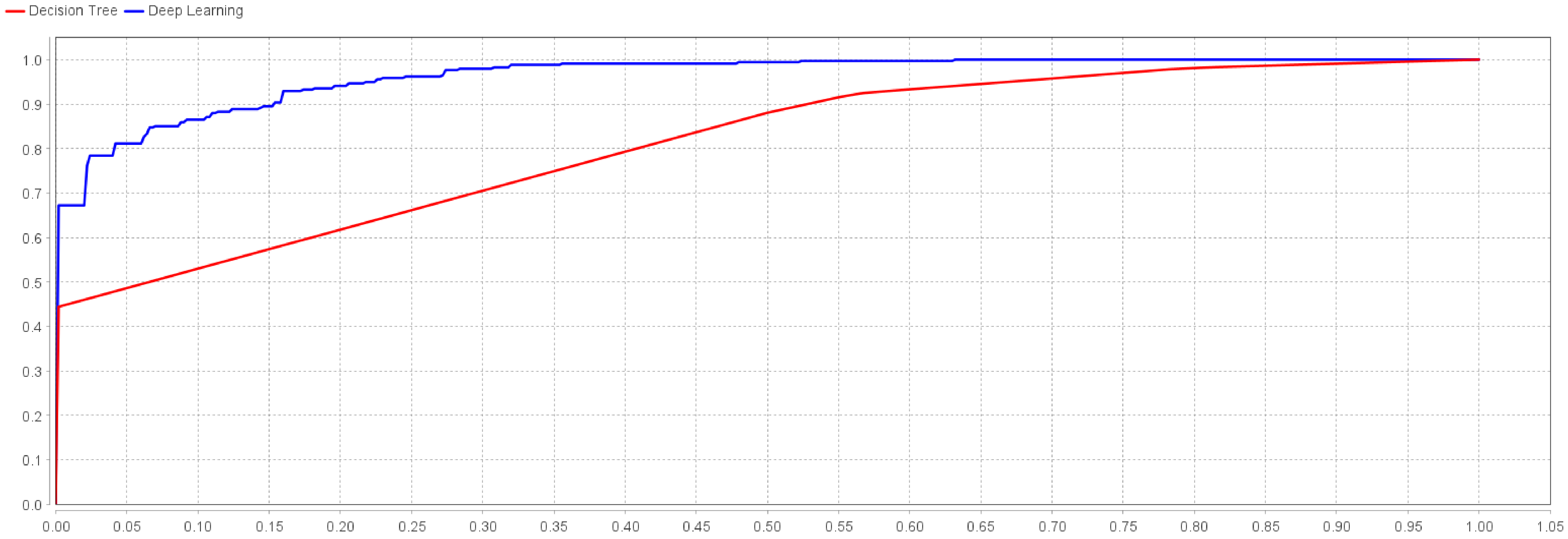
Results

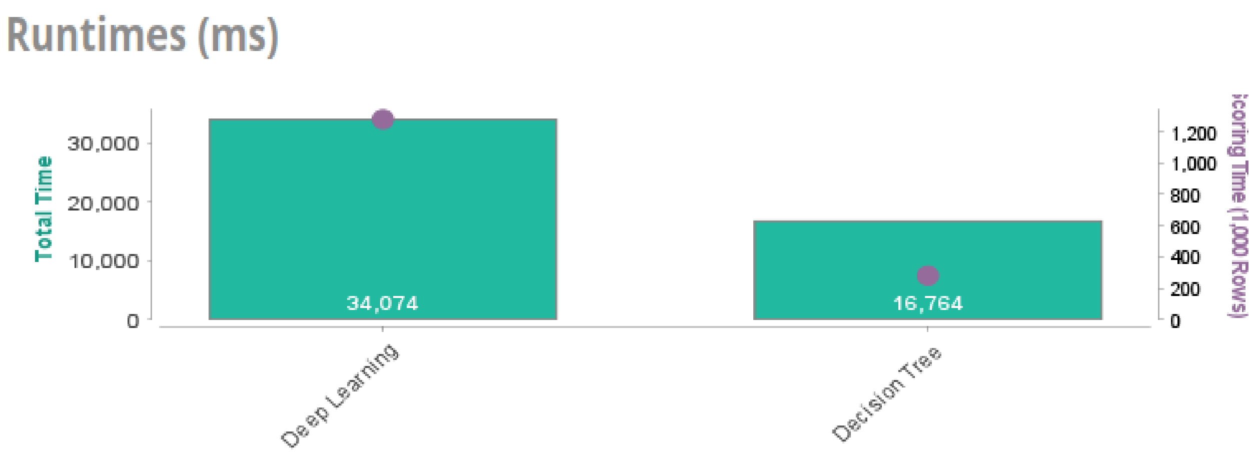
Row No.	Delay	prediction(D...	confidence(Y)	confidence(N)	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime
1	?	N	0.202	0.798	1	30	6	?	1920	2016
2	?	Y	1	0	2	6	6	?	728	957
3	?	N	0.202	0.798	3	17	3	?	0	1330
4	?	N	0.202	0.798	4	24	6	?	700	920
5	?	N	0.202	0.798	5	1	7	?	1205	1306
6	?	N	0.202	0.798	6	7	2	?	945	1044
7	?	Y	1	0	7	14	4	?	1455	1730
8	?	N	0.202	0.798	8	22	1	?	1750	1938
9	?	N	0.202	0.798	9	12	3	?	1500	1602
10	?	Y	1	0	10	14	7	?	1500	1715
11	?	Y	1	0	11	6	2	?	730	1022
12	?	N	0.202	0.798	12	14	5	?	2030	2130

accuracy: 94.50% +/- 7.78% (micro average: 94.50%)

	true Y	true N	class precision
pred. Y	1058	0	100.00%
pred. N	110	832	88.32%
class recall	90.58%	100.00%	

Results





Accuracy ▼

Model	Accuracy	Standard Deviation	Gains	Total Time	Training Time (1,000 ...	Scoring Time (1,0
Deep Learning	59.1%	± 1.7%	0	34 s	928 ms	1 s
Decision Tree	66.1%	± 1.4%	94	17 s	49 ms	279 ms

Parth Dodia

Feature Preprocessing

	Name	Description
1	Year	1987-2008
2	Month	1-12
3	DayofMonth	1-31
4	DayOfWeek	1 (Monday) - 7 (Sunday)
5	DepTime	actual departure time (local, hhmm)
6	CRSDepTime	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	UniqueCarrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code
18	Dest	destination IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 30 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Year               2000 non-null   int64
 1   Month              2000 non-null   int64
 2   DayofMonth         2000 non-null   int64
 3   DayOfWeek          2000 non-null   int64
 4   DepTime            2000 non-null   float64
 5   CRSDepTime         2000 non-null   int64
 6   ArrTime            2000 non-null   float64
 7   CRSArrTime         2000 non-null   int64
 8   UniqueCarrier      2000 non-null   object
 9   FlightNum          2000 non-null   int64
 10  TailNum            2000 non-null   object
 11  ActualElapsedTime  2000 non-null   float64
 12  CRSElapsedTime     2000 non-null   float64
 13  AirTime            2000 non-null   float64
 14  ArrDelay           2000 non-null   float64
 15  DepDelay           2000 non-null   float64
 16  Origin             2000 non-null   object
 17  Dest               2000 non-null   object
 18  Distance           2000 non-null   int64
 19  TaxiIn             2000 non-null   int64
 20  TaxiOut            2000 non-null   int64
 21  Cancelled          2000 non-null   int64
 22  CancellationCode   0 non-null      float64
 23  Diverted           2000 non-null   int64
 24  CarrierDelay       0 non-null      float64
 25  WeatherDelay       0 non-null      float64
 26  NASDelay           0 non-null      float64
 27  SecurityDelay      0 non-null      float64
 28  LateAircraftDelay  0 non-null      float64
 29  Delay              2000 non-null   object
dtypes: float64(13), int64(12), object(5)
memory usage: 468.9+ KB
```

Feature Preprocessing

```
def convert_hhmm_m(df, columns_list: list):
    '''This function converts time in hhmm to mm : hh * 60 + mm
    Input: df -> pandas data frame
    columns_list: list of columns to be converted
    '''

    for column in columns_list:
        df.loc[:,column] = ((df.loc[:,column]//100)*60 + (df.loc[:,column]%100)).div(60)
```

.1s

```
columns_list = ['CRSDepTime', 'CRSArrTime', 'ArrTime']
convert_hhmm_m(df= df_sample, columns_list=columns_list)
```

Feature Preprocessing

```python
from feature_engine.creation import CyclicalFeatures
columns_list1 = ['Month', 'DayofMonth','DayOfWeek']
cf = CyclicalFeatures(variables=columns_list+columns_list1, drop_original=True)
time_features = df_sample[columns_list+columns_list1]
df_t = cf.fit_transform(time_features)
```

.1s Python

```python
df_t.head()
```

.1s Python

RSDepTime_sin	CRSDepTime_cos	CRSArrTime_sin	CRSArrTime_cos	ArrTime_sin	ArrTime_cos	Month_sin	Month_cos	DayofMonth_sin	DayofMonth_cos	DayOfWeek_sin	DayOfWeek_cos
-0.994628	0.103515	-0.740607	0.671938	-0.737277	0.675590	-1.000000e+00	-1.836970e-16	0.998717	-0.050649	0.433884	-0.900969
-0.933097	-0.359624	-0.987842	0.155464	-0.981627	0.190809	8.660254e-01	-5.000000e-01	0.651372	-0.758758	0.781831	0.623490
0.792411	-0.609988	0.239968	-0.970781	0.224951	-0.974370	-2.449294e-16	1.000000e+00	0.299363	-0.954139	0.974928	-0.222521
0.886352	-0.463012	0.535790	-0.844351	0.540974	-0.841039	1.000000e+00	6.123234e-17	-0.485302	-0.874347	-0.433884	-0.900969
0.842592	-0.538552	0.674360	-0.738403	0.700909	-0.713250	-8.660254e-01	-5.000000e-01	-0.571268	0.820763	-0.781831	0.623490

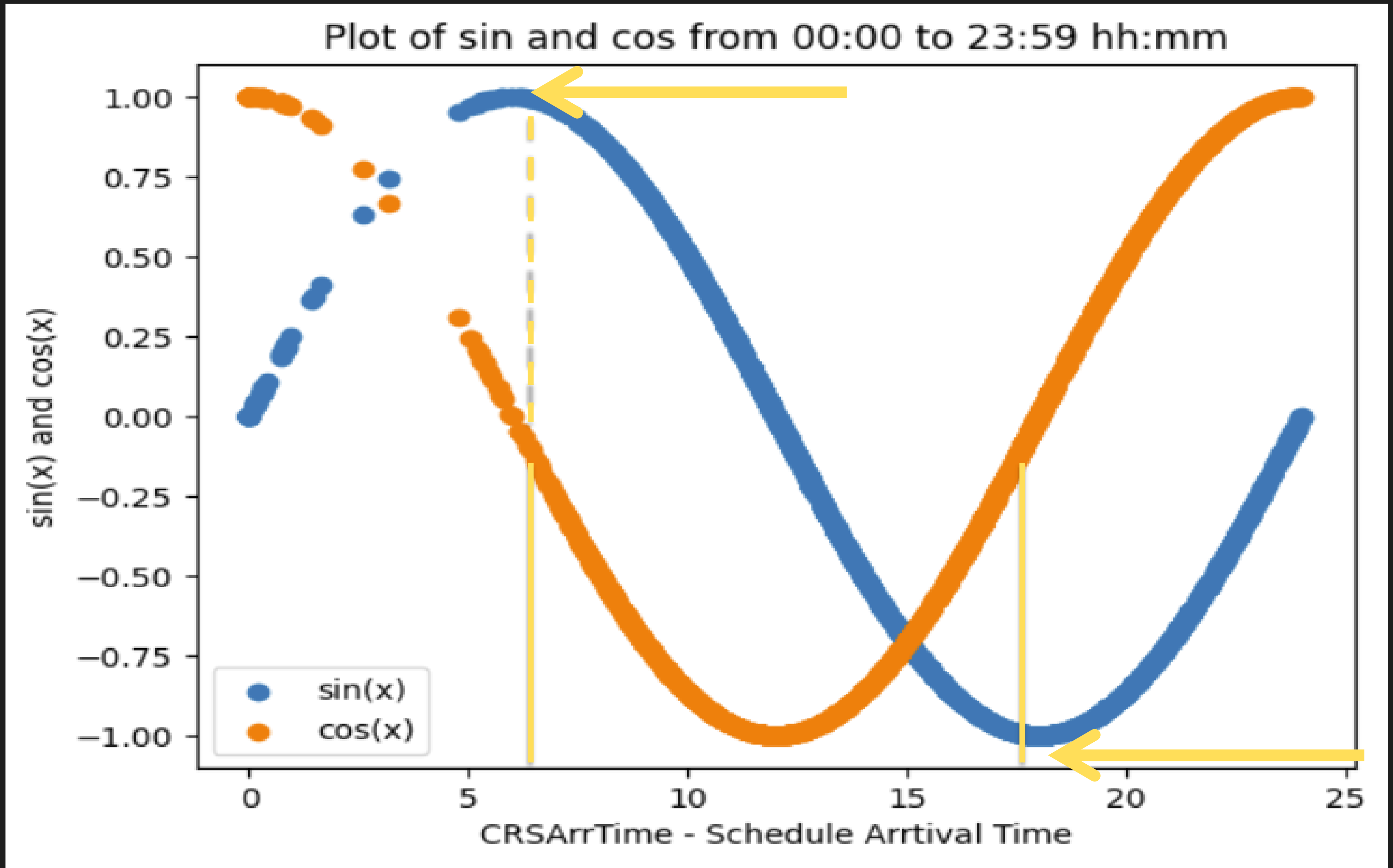
Feature Preprocessing

```
import matplotlib.pyplot as plt
```

✓ 0.1s

```
plt.scatter(df_sample['CRSArrTime'], df_t['CRSArrTime_sin'])
plt.scatter(df_sample['CRSArrTime'], df_t['CRSArrTime_cos'])
plt.xlabel('CRSArrTime - Schedule Arrtival Time')  # string must be enclosed with quotes '  '
plt.ylabel('sin(x) and cos(x)')
plt.title('Plot of sin and cos from 00:00 to 23:59 hh:mm')
plt.legend(['sin(x)', 'cos(x)'])
plt.show()
```

✓ 0.3s



Feature Preprocessing

```
df_final = pd.concat([df_sample,df_t],axis=1)
```

✓ 0.0s

```
df_final = df_final.drop(columns=['Month', 'DayofMonth','DayOfWeek', 'CRSDepTime', 'CRSArrTime', 'ArrTime',
                                  'Cancelled', 'FlightNum', 'TailNum', 'CancellationCode', 'Diverted', 'CarrierDelay', 'WeatherDelay', 'NASDelay',
                                 'SecurityDelay', 'LateAircraftDelay'])
```

✓ 0.1s

```
df_final.columns
```

✓ 0.0s

```
Index(['Year', 'DepTime', 'UniqueCarrier', 'ActualElapsedTime',
       'CRSElapsedTime', 'AirTime', 'ArrDelay', 'DepDelay', 'Origin', 'Dest',
       'Distance', 'TaxiIn', 'TaxiOut', 'Delay', 'CRSDepTime_sin',
       'CRSDepTime_cos', 'CRSArrTime_sin', 'CRSArrTime_cos', 'ArrTime_sin',
       'ArrTime_cos', 'Month_sin', 'Month_cos', 'DayofMonth_sin',
       'DayofMonth_cos', 'DayOfWeek_sin', 'DayOfWeek_cos'],
      dtype='object')
```

```
df_final.to_csv('sample_1999+2000_time.csv',index=False)
```

✓ 0.1s

```
df_final.shape
```

✓ 0.0s

```
(2000, 26)
```

Feature Preprocessing

```
1 df_test
```

✓ 0.0s

CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance	TaxiIn	TaxiOut
2030	US	1541	N275AU	NaN	70	42.0	-14	NaN	ROC	PHL	257	8.0	13.0
944	TW	606	N921L	NaN	76	53.0	13	NaN	STL	TYS	405	4.0	19.0
0	UA	1660	N951UA	NaN	205	182.0	-10	NaN	DEN	PHL	1557	3.0	13.0
940	UA	462	N998UA	NaN	100	74.0	-20	NaN	ORD	CLT	599	2.0	8.0
1312	TW	447	NaN	NaN	127	NaN	-6	NaN	ORF	STL	784	NaN	NaN
1048	US	253	NaN	NaN	63	NaN	-4	NaN	PIT	LEX	289	NaN	NaN
1714	CO	1098	NaN	NaN	139	NaN	16	NaN	IAH	ORD	925	NaN	NaN
1942	DL	922	NaN	NaN	112	NaN	-4	NaN	CVG	LGA	585	NaN	NaN
1605	OO	3691	N565SW	NaN	65	52.0	-3	NaN	SUN	SLC	223	4.0	8.0
1710	MQ	4471	N820AE	NaN	130	109.0	5	NaN	ORD	PNS	794	3.0	18.0
1013	NW	1674	N613NW	NaN	163	147.0	9	NaN	AUS	MSP	1042	11.0	20.0
2144	DL	1969	N916DE	NaN	74	40.0	-14	NaN	LGA	DCA	214	4.0	17.0

```
1 # ActualElapsedTime  = AirTime + TaxiIn + TaxiOut
2 df_test['ActualElapsedTime'] = df_test['AirTime'] + df_test['TaxiIn'] + df_test['TaxiOut']
```

✓ 0.0s

Feature Preprocessing

ne	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance	TaxiIn
20	2016	2030	US	1541	N275AU	63.0	70	42.0	-14	NaN	ROC	PHL	257	8.0
28	957	944	TW	606	N921L	76.0	76	53.0	13	NaN	STL	TYS	405	4.0
0	1330	0	UA	1660	N951UA	198.0	205	182.0	-10	NaN	DEN	PHL	1557	3.0
00	920	940	UA	462	N998UA	84.0	100	74.0	-20	NaN	ORD	CLT	599	2.0
05	1306	1312	TW	447	NaN	NaN	127	NaN	-6	NaN	ORF	STL	784	NaN
45	1044	1048	US	253	NaN	NaN	63	NaN	-4	NaN	PIT	LEX	289	NaN
55	1730	1714	CO	1098	NaN	NaN	139	NaN	16	NaN	IAH	ORD	925	NaN
50	1938	1942	DL	922	NaN	NaN	112	NaN	-4	NaN	CVG	LGA	585	NaN
00	1602	1605	OO	3691	N565SW	64.0	65	52.0	-3	NaN	SUN	SLC	223	4.0
00	1715	1710	MQ	4471	N820AE	130.0	130	109.0	5	NaN	ORD	PNS	794	3.0
30	1022	1013	NW	1674	N613NW	178.0	163	147.0	9	NaN	AUS	MSP	1042	11.0
30	2130	2144	DL	1969	N916DE	61.0	74	40.0	-14	NaN	LGA	DCA	214	4.0

```python
# ArrDelay = ArrTime - CRSArrTime
df_test.loc[2,'CRSArrTime'] = df_test.loc[2,'ArrTime'] - df_test.loc[2,'ArrDelay']
```

✓ 0.0s Python

Feature Preprocessing

```python
# CRSElapsedTime = CRSArrTime - CRSDepTime
CRSArrTime = 820 # 13*60+40 (13hh:40mm)
CRSElapsedTime = 205
diff = CRSArrTime - CRSElapsedTime # 615
diff = (diff//60)*100 + diff%60
df_test.loc[2,'CRSDepTime'] = diff
```

✓ 0.0s

```python
df_test
```

✓ 0.1s

DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	DepDelay
NaN	1920	2016	2030	US	1541	N275AU	63.0	70	42.0	-14	NaN
NaN	728	957	944	TW	606	N921L	76.0	76	53.0	13	NaN
NaN	1015	1330	1340	UA	1660	N951UA	198.0	205	182.0	-10	NaN
NaN	700	920	940	UA	462	N998UA	84.0	100	74.0	-20	NaN
NaN	1205	1306	1312	TW	447	NaN	NaN	127	NaN	-6	NaN
NaN	945	1044	1048	US	253	NaN	NaN	63	NaN	-4	NaN
NaN	1455	1730	1714	CO	1098	NaN	NaN	139	NaN	16	NaN
NaN	1750	1938	1942	DL	922	NaN	NaN	112	NaN	-4	NaN
NaN	1500	1602	1605	OO	3691	N565SW	64.0	65	52.0	-3	NaN
NaN	1500	1715	1710	MQ	4471	N820AE	130.0	130	109.0	5	NaN
NaN	730	1022	1013	NW	1674	N613NW	178.0	163	147.0	9	NaN
[illegible]	2030	2130	2144	DL	1969	N916DE	61.0	74	40.0	-14	NaN

Feature Preprocessing

```python
columns_list = ['CRSDepTime', 'CRSArrTime', 'ArrTime']
convert_hhmm_m(df= df_test, columns_list=columns_list)
```

✓ 0.0s

```python
columns_list1 = ['Month', 'DayofMonth','DayOfWeek']
featues_changes = df_test[columns_list+columns_list1]
df_trans_test = cf.transform(featues_changes)
```

✓ 0.0s

```python
df_test_final = pd.concat([df_test,df_trans_test],axis=1)
df_test_final = df_test_final.drop(columns=['Month', 'DayofMonth','DayOfWeek', 'CRSDepTime', 'CRSArrTime', 'ArrTime',
                                            'Cancelled', 'FlightNum', 'TailNum', 'CancellationCode', 'Diverted', 'CarrierDelay', 'WeatherDelay', 'NASDelay',
                                           'SecurityDelay', 'LateAircraftDelay'])
```

✓ 0.1s

```python
df_test_final.shape
```

✓ 0.0s

```
(12, 26)
```

```python
df_test_final.columns
```

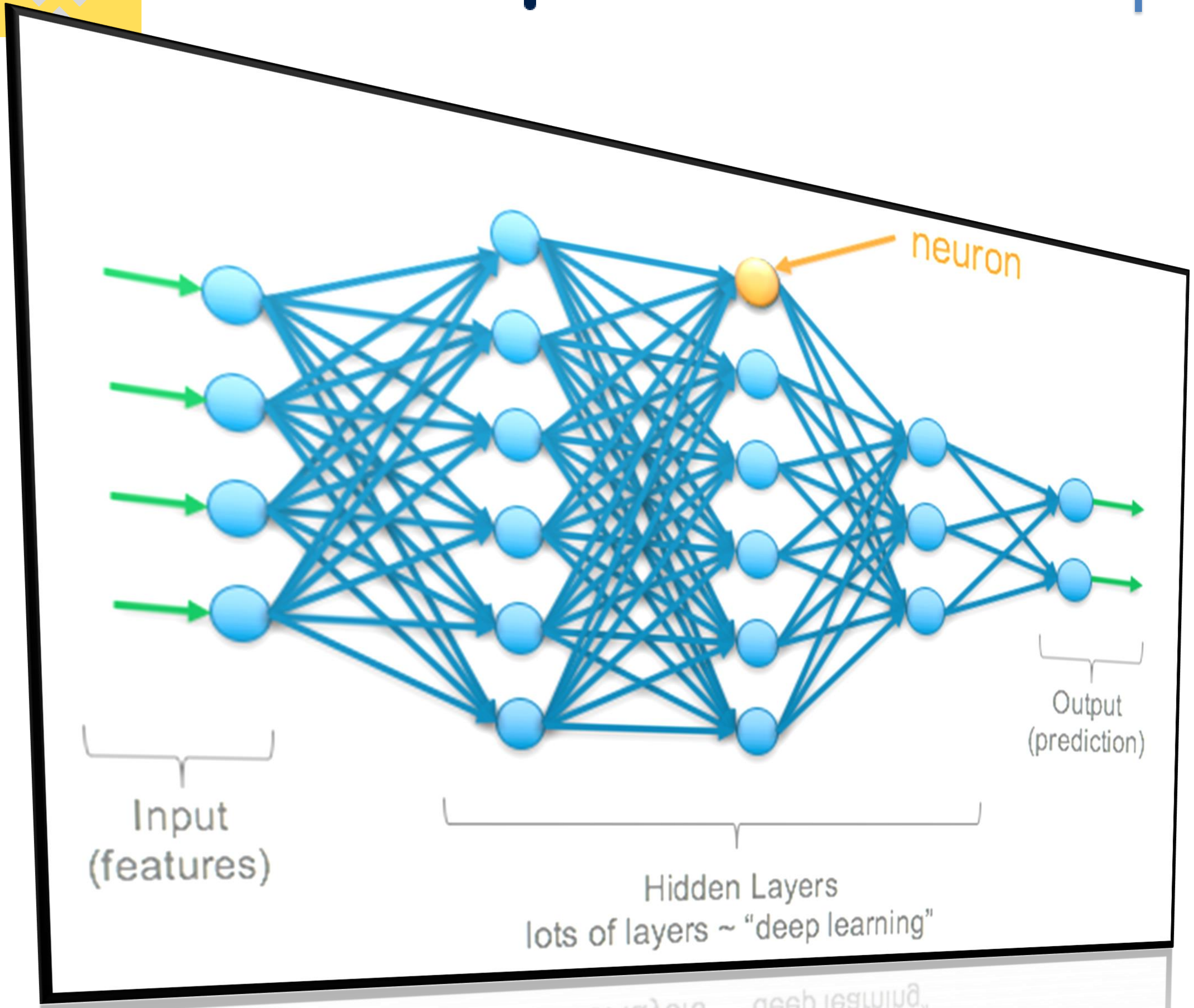
✓ 0.0s

```
Index(['Year', 'DepTime', 'UniqueCarrier', 'ActualElapsedTime',
       'CRSElapsedTime', 'AirTime', 'ArrDelay', 'DepDelay', 'Origin', 'Dest',
       'Distance', 'TaxiIn', 'TaxiOut', 'Delayed (Y or N)', 'CRSDepTime_sin',
       'CRSDepTime_cos', 'CRSArrTime_sin', 'CRSArrTime_cos', 'ArrTime_sin',
       'ArrTime_cos', 'Month_sin', 'Month_cos', 'DayofMonth_sin',
       'DayofMonth_cos', 'DayOfWeek_sin', 'DayOfWeek_cos'],
      dtype='object')
```

```python
df_test_final.to_csv('sample_12_records.csv',index=False)
```

✓ 0.0s

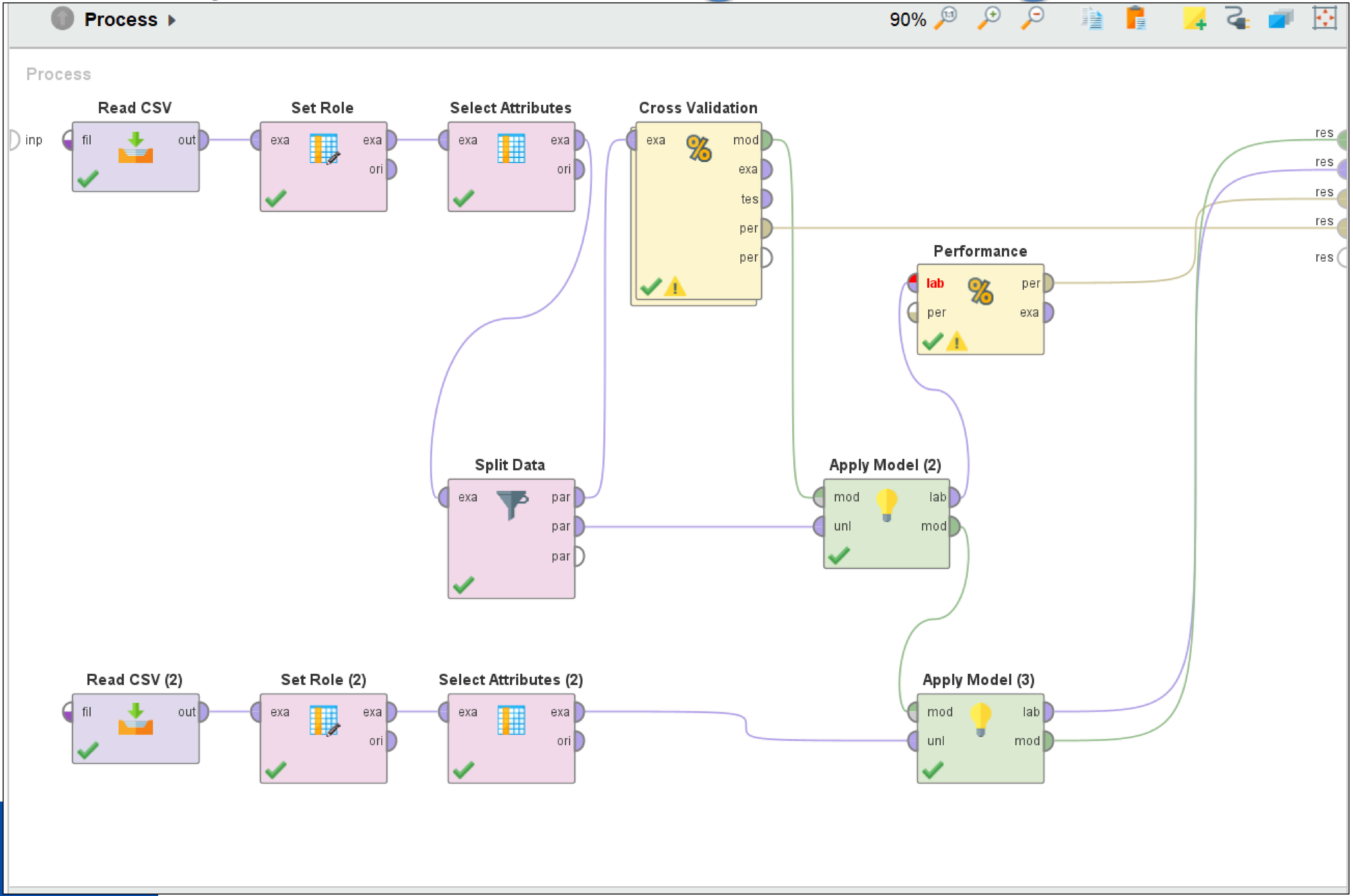
Model – Deep Learning



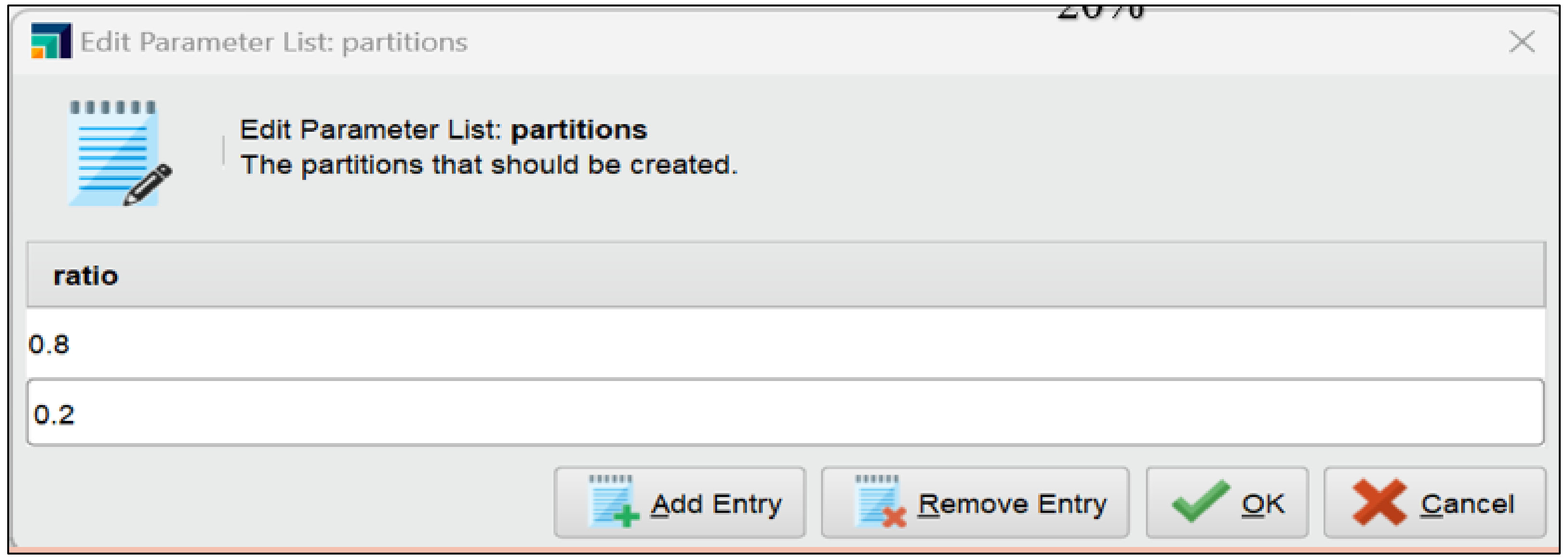
Deep learning is a subfield of machine learning that involves the use of artificial neural networks to model and solve complex problems.

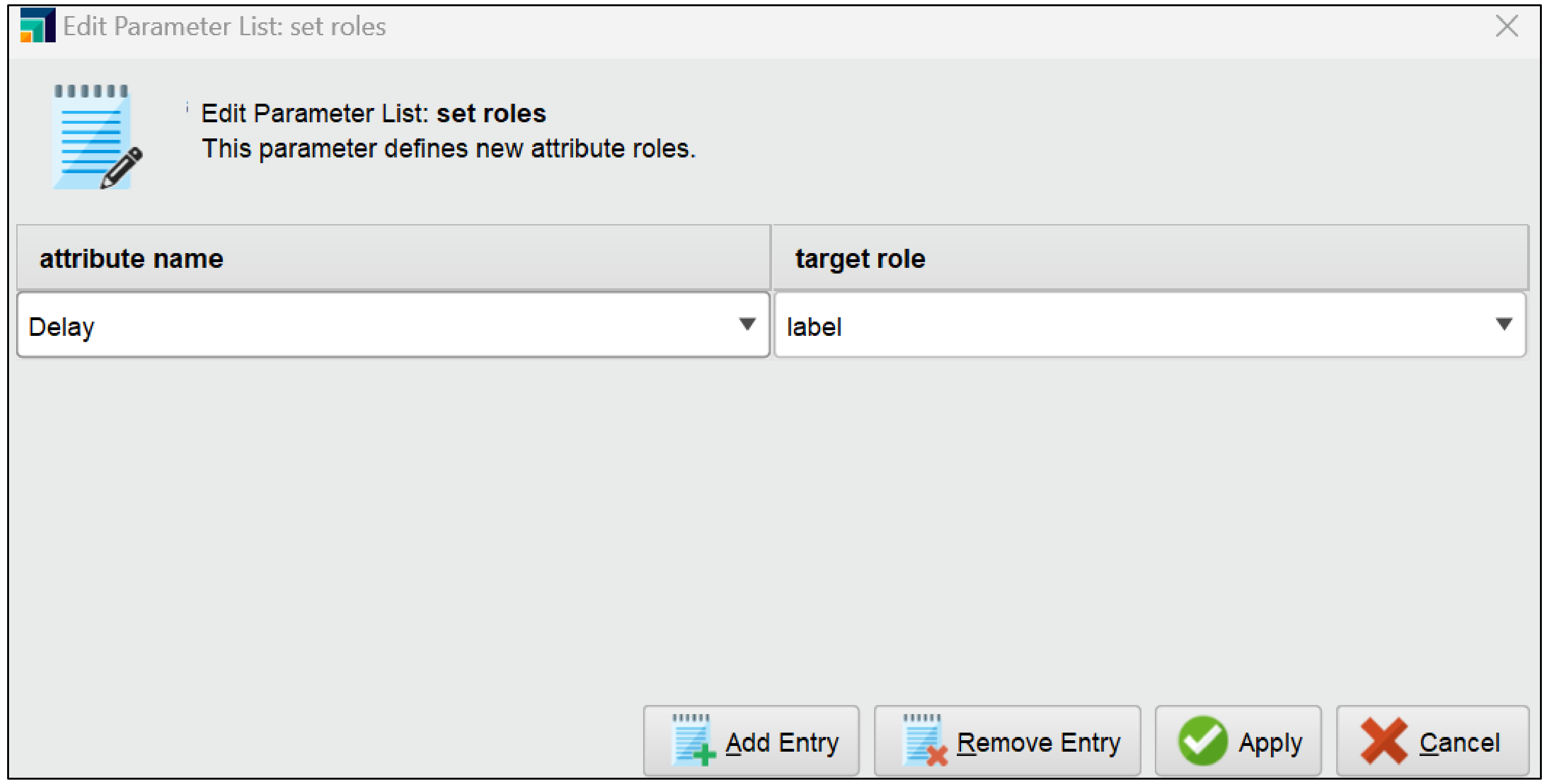
Deep learning algorithms are designed to learn multiple levels of representations of data, and they have proven to be highly effective in tasks such as prediction, image recognition, natural language processing, speech recognition, and many others.

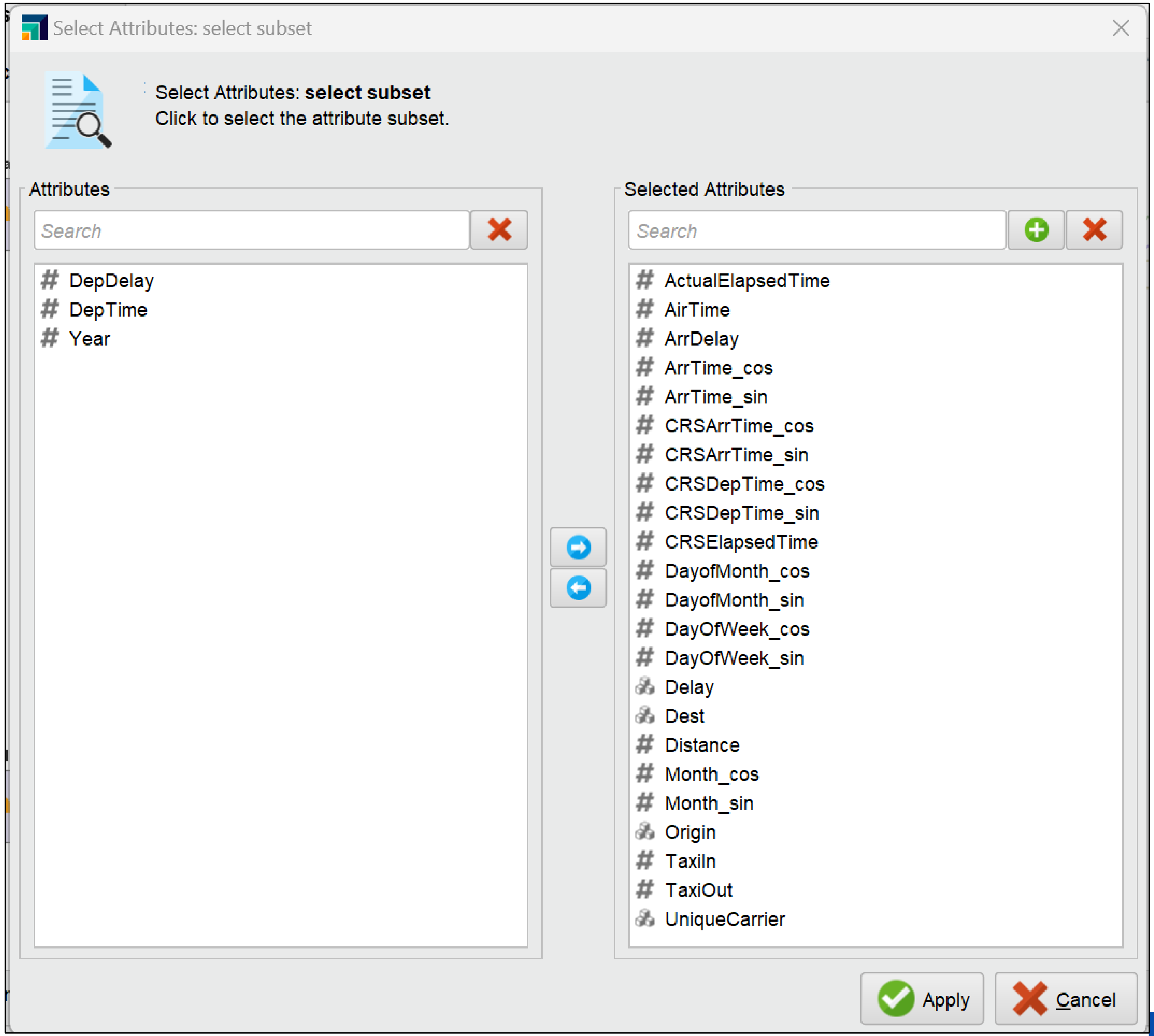
Model Training & Testing



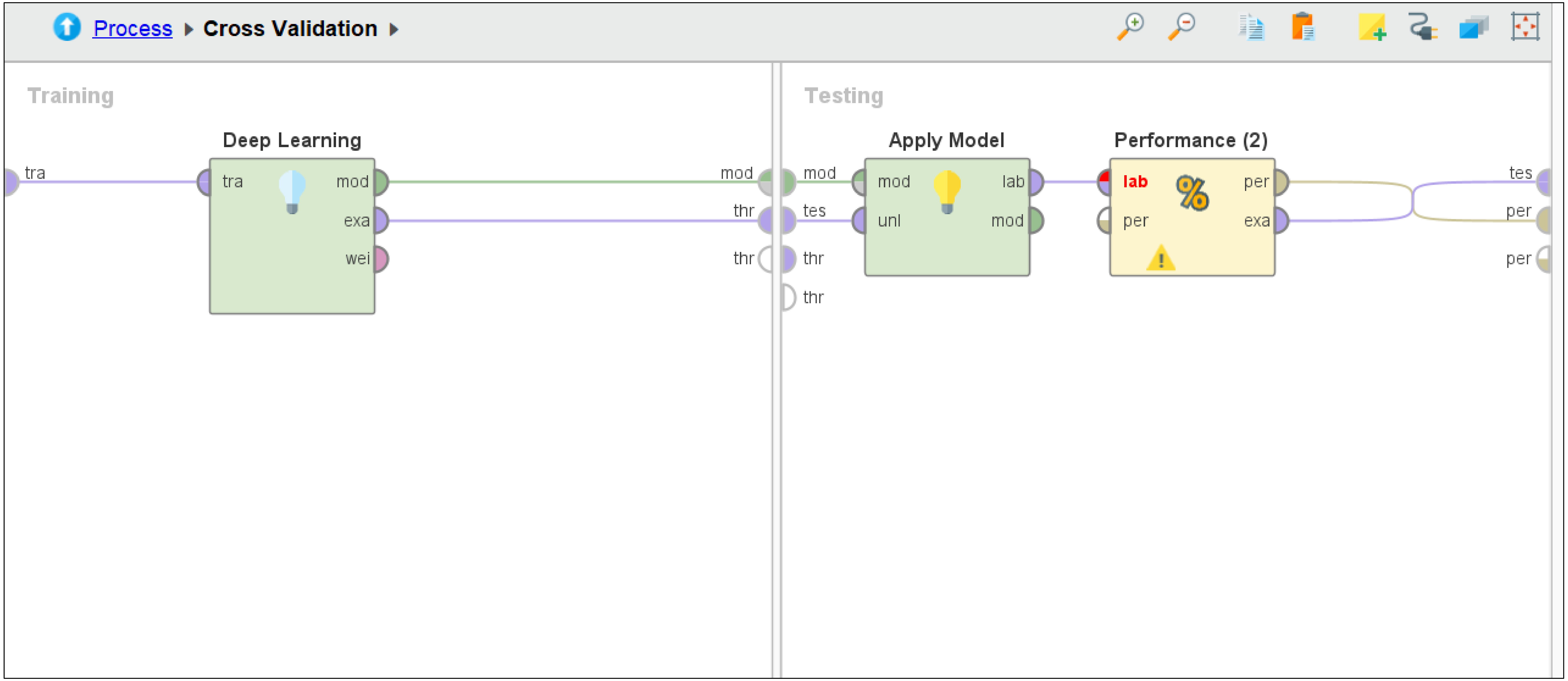
Model Training & Testing







Model Training & Testing



Yash Jivani

Model Training & Testing

Performance Score – Training Set

accuracy: 87.25% +/- 6.28% (micro average: 87.25%)

	true Y	true N	class precision
pred. Y	799	44	94.78%
pred. N	160	597	78.86%
class recall	83.32%	93.14%	

Performance Score – Test Set

accuracy: 91.00%

	true Y	true N	class precision
pred. Y	209	5	97.66%
pred. N	31	155	83.33%
class recall	87.08%	96.88%	

Model Training & Testing

Row No.	Delayed (Y ...	prediction(D...	confidence(...	confidence(...	UniqueCarri...	ActualElaps...	CRSElapse...	AirTime	ArrDelay	Origin	Dest
1	?	N	0.009	0.991	US	63	70	42	-14	ROC	PHL
2	?	Y	1.000	0.000	TW	76	76	53	13	STL	TYS
3	?	N	0.049	0.951	UA	198	205	182	-10	DEN	PHL
4	?	N	0.009	0.991	UA	84	100	74	-20	ORD	CLT
5	?	Y	0.997	0.003	TW	?	127	?	-6	ORF	STL
6	?	N	0.434	0.566	US	?	63	?	-4	PIT	LEX
7	?	Y	1.000	0.000	CO	?	139	?	16	IAH	ORD
8	?	Y	0.997	0.003	DL	?	112	?	-4	CVG	LGA
9	?	N	0.093	0.907	OO	64	65	52	-3	SUN	SLC
10	?	Y	0.997	0.003	MQ	130	130	109	5	ORD	PNS
11	?	Y	1.000	0.000	NW	178	163	147	9	AUS	MSP
12	?	N	0.013	0.987	DL	61	74	40	-14	LGA	DCA

Total predicted Delay(Y) = 6

Total predicted Delay(N) = 6

Conclusion

- The accuracy from decision tree model was found to be 94.5% while from deep learning model it is 91%.
- In real world, data pre-processing is more important than just getting higher accuracy to predict unseen records.
- When input features are cyclic (i.e. day, month, time, etc), the model can perform intilally well without transforming the features but perform worst during prediction
- In the end we conclude that deep learning model with pre-processing is better than decision tree.

THANK YOU FOR YOUR TIME