DEVELOPING A FLIGHT DELAY MODEL USING MACHINE LEARNING

TEAM ID: PNT2022TMID27775

Label Encoding & One Hot Encoding

- Typically, any structured dataset includes multiple columns with combinations of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with Machine Learning algorithms too. We need to convert each text category to numbers in order for the machine to process those using mathematical equations.
- How should we handle categorical variables? There are Multiple ways to handle it, but I will see one of them is Label Encoding.
- Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.
- Let's see how to implement label encoding in Python using the scikit-learn library.
- As we have to convert only the text class category columns, we first select it then we will implement Label Encoding to it.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['DEST'] = le.fit_transform(dataset['DEST'])
dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

```
dataset.head(5)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	0	4	21	0.0	0.0
1	1476	1	1	5	1	3	14	0.0	0.0
2	1597	1	1	5	0	4	12	0.0	0.0
3	1768	1	1	5	4	3	13	0.0	0.0
4	1823	1	1	5	4	1	6	0.0	0.0

- In the above code we are looping through all the selected text class categorical columns and performing label encoding.
- If you see output of the above code, after performing label encoding alphabetical classes is converted to numeric.

- The most popular way to encode nominal features is one-hot-encoding. Essentially, each categorical feature with n categories is transformed into n binary features.
- Let's take a look at our example to make things clear. Start with importing the OneHotEncoder class and creating a new instance with the output data type set to integer. This doesn't change anything to how our data will be interpreted, but will improve the readability of our output.
- Then, fit and transform our two nominal categoricals. The output of this transformation will be a sparse matrix, this means we'll have to transform the matrix into an array (.toarray()) before we can pour it into a dataframe. You can omit this step by setting the sparse parameter to False when initiating a new class instance.

```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder()
z=oh.fit_transform(x[:,4:5]).toarray()
t=oh.fit_transform(x[:,5:6]).toarray()
```

```
z
```

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       ...,
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.]])
```

```
t
```

```
array([[0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.],
       ...,
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0.]])
```