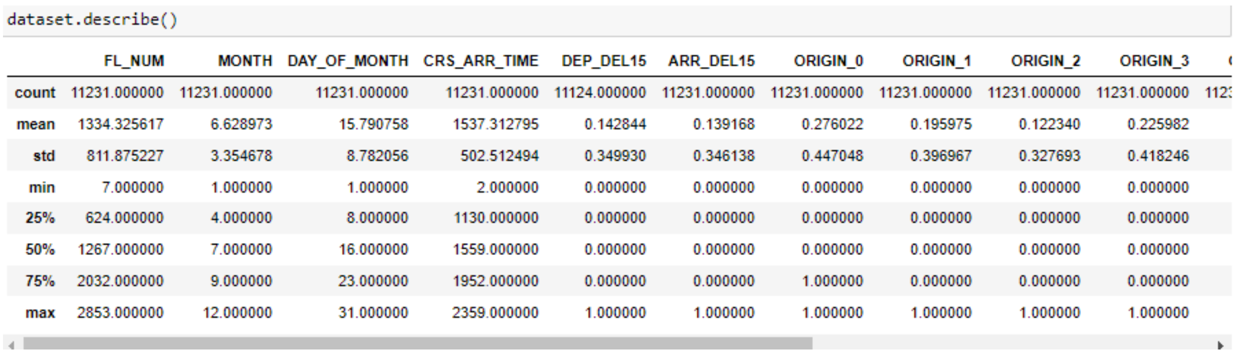
## Descriptive Statistical:

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
dataset.describe()
```

| | FL_NUM | MONTH | DAY_OF_MONTH | CRS_ARR_TIME | DEP_DEL15 | ARR_DEL15 | ORIGIN_0 | ORIGIN_1 | ORIGIN_2 | ORIGIN_3 |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11124.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 |
| **mean** | 1334.325617 | 6.628973 | 15.790758 | 1537.312795 | 0.142844 | 0.139168 | 0.276022 | 0.195975 | 0.122340 | 0.225982 |
| **std** | 811.875227 | 3.354678 | 8.782056 | 502.512494 | 0.349930 | 0.346138 | 0.447048 | 0.396967 | 0.327693 | 0.418246 |
| **min** | 7.000000 | 1.000000 | 1.000000 | 2.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 624.000000 | 4.000000 | 8.000000 | 1130.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **50%** | 1267.000000 | 7.000000 | 16.000000 | 1559.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **75%** | 2032.000000 | 9.000000 | 23.000000 | 1952.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **max** | 2853.000000 | 12.000000 | 31.000000 | 2359.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

## Visual Analysis

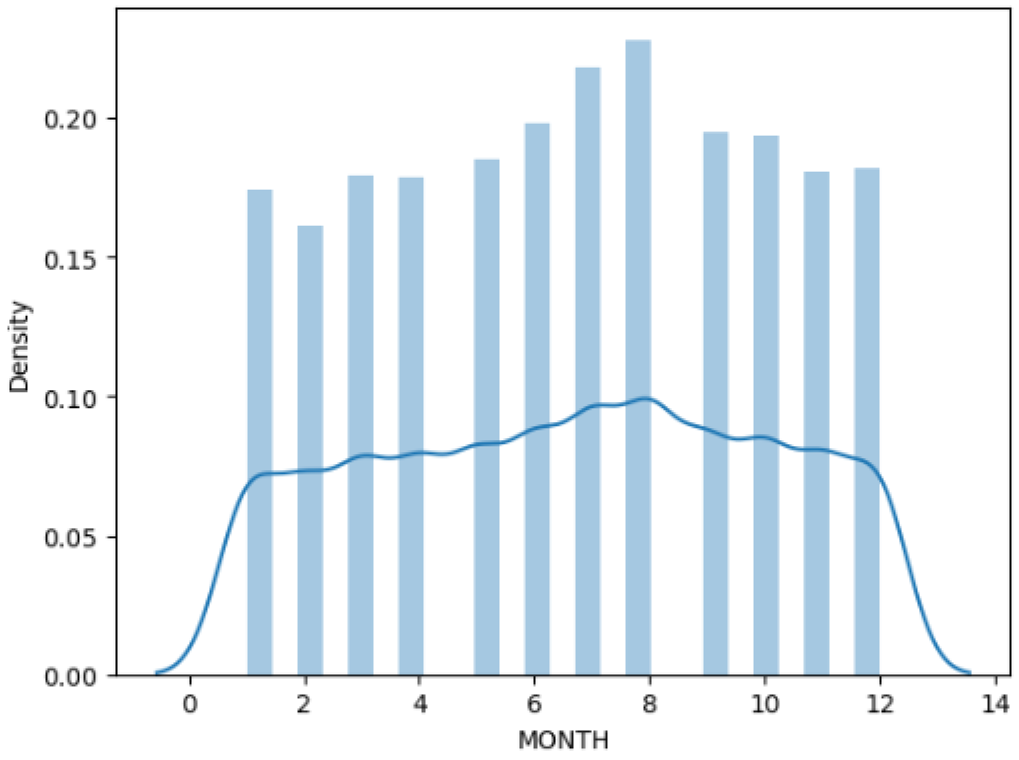
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

## Univariate Analysis

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
<Axes: xlabel='MONTH', ylabel='Density'>
```



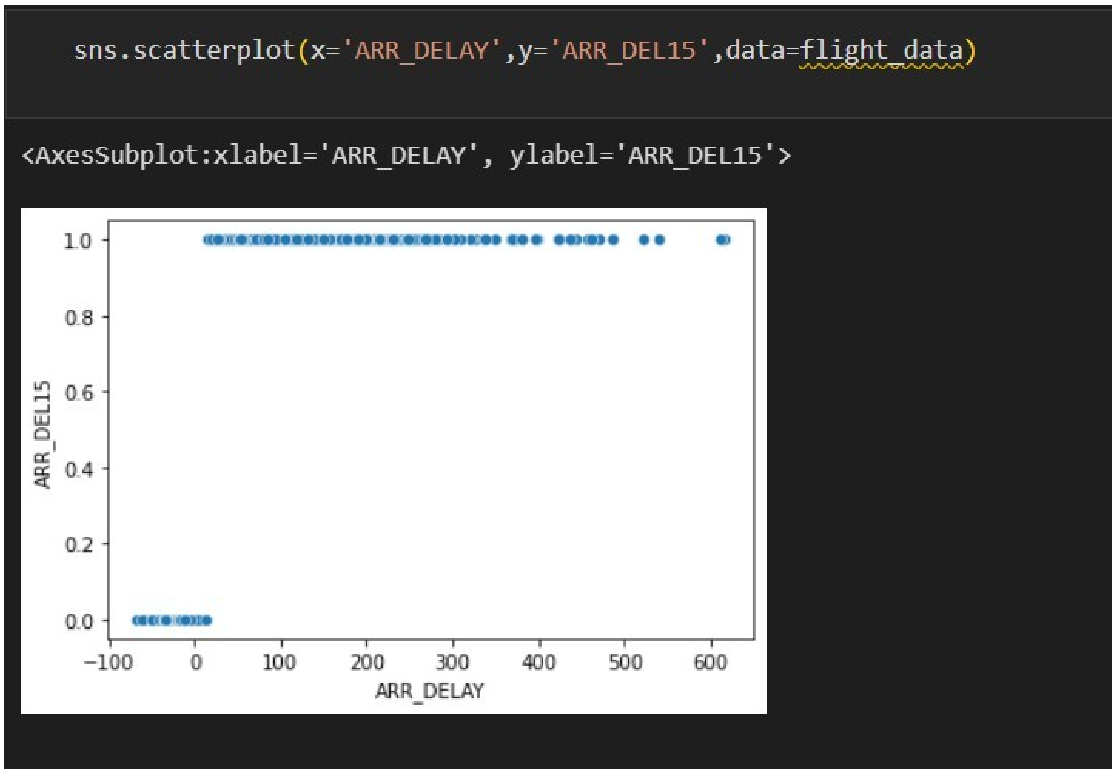
- In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.
- From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0

**Countplot:-**

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

From the graph we can infer that , gender and education is a categorical variables with 2 categories , from gender column we can infer that 0-category is having more weightage than category-1,while education with 0,it means no education is a underclass when compared with category -1, which means educated .

## Bivariate Analysis



```
sns.scatterplot(x='ARR_DELAY',y='ARR_DEL15',data=flight_data)
```

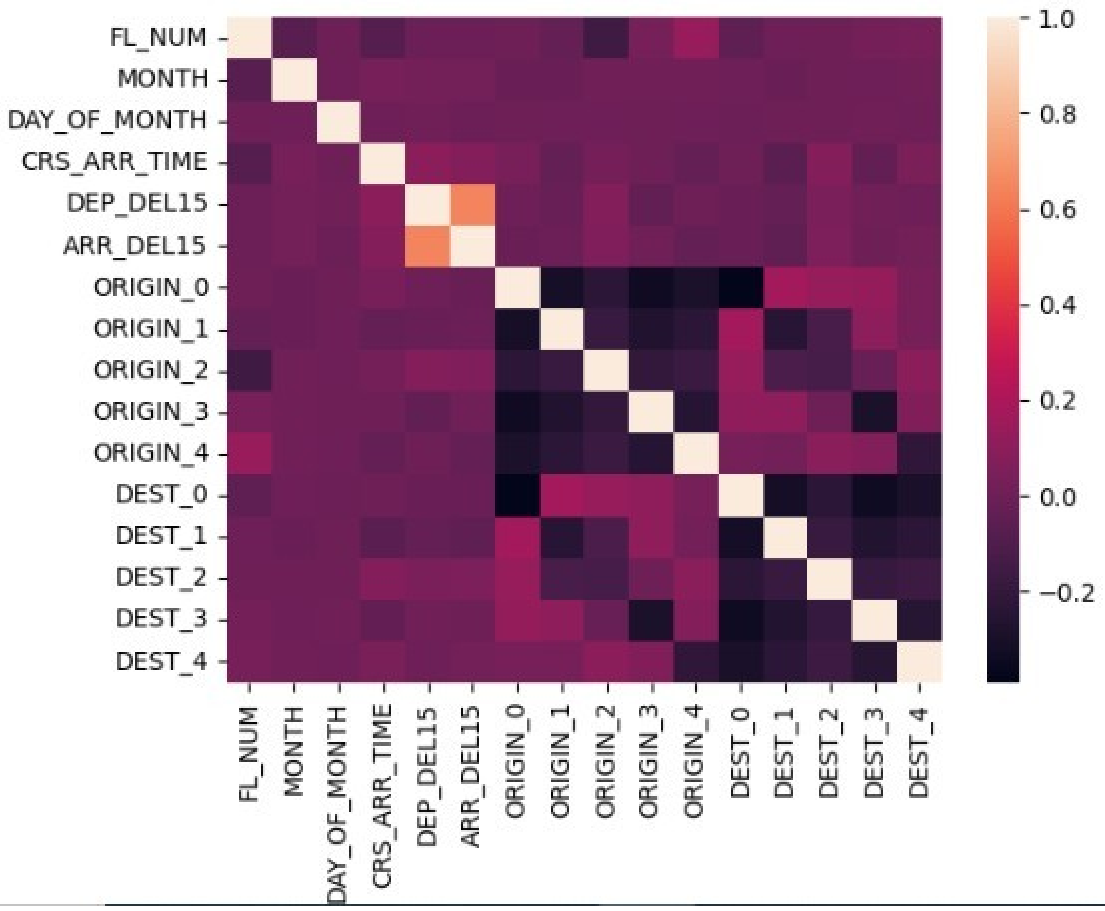
```
<AxesSubplot:xlabel='ARR_DELAY', ylabel='ARR_DEL15'>
```

## Multivariate Analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a swarm plot from the seaborn package.

```
sns.heatmap(dataset.corr())
```

```
<Axes: >
```



From the above graph we are plotting the relationship all the features.

**Splitting data into dependent and independent variables**

```
dataset = pd.get_dummies(dataset, columns=['ORIGIN', 'DEST'])
dataset.head()
```

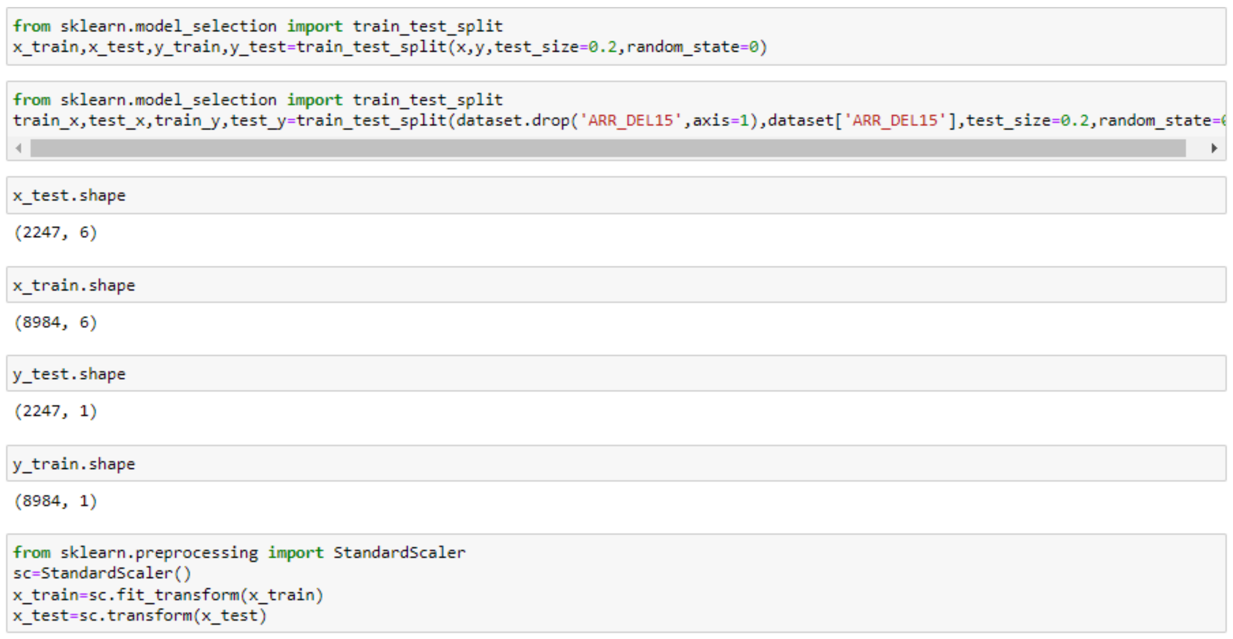
```
x = dataset.iloc[:, 0:8].values
y = dataset.iloc[:, 8:9].values
```

## Splitting data into train and test & Scaling the data:

Now let’s split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y=train_test_split(dataset.drop('ARR_DEL15',axis=1),dataset['ARR_DEL15'],test_size=0.2,random_state=
```

```
x_test.shape
```

```
(2247, 6)
```

```
x_train.shape
```

```
(8984, 6)
```

```
y_test.shape
```

```
(2247, 1)
```

```
y_train.shape
```

```
(8984, 1)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```