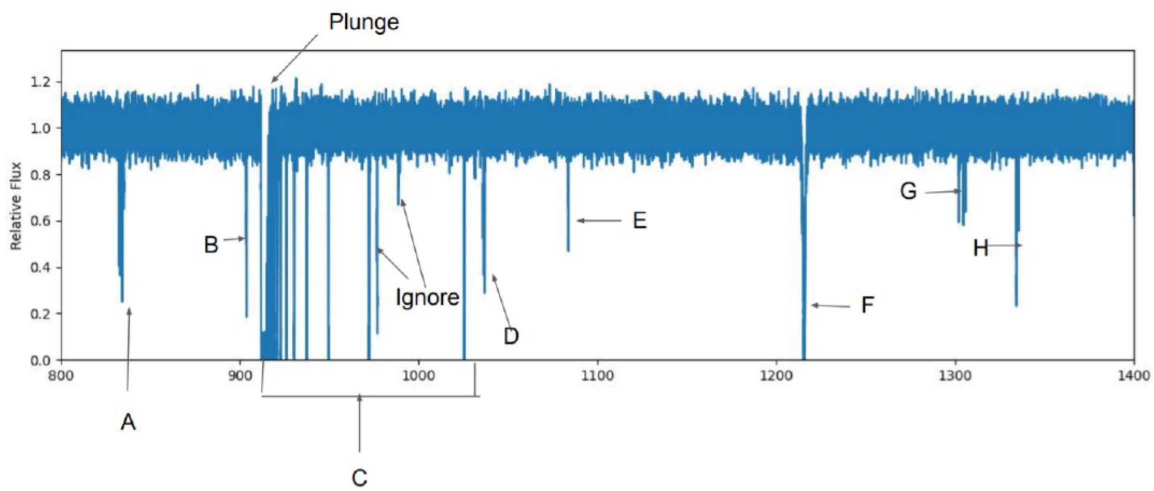
Task 1: Ion Identification



A. For features A-H the corresponding atoms and ions are:
   - A → O II
   - B → C II
   - C → H I
   - D → O I
   - E → N II
   - F → H I
   - G → O I
   - H → C II

B. The features in C belong to the Lyman Series of Hydrogen atoms. The plunge near 900 Å represents the dark lines in the absorption spectrum which correspond to the frequencies of light that have been absorbed by the gas.

The following gives the wavelength of radiation absorbed /emitted when electron transitions between two states:

$$\nu = c/\lambda = \Delta E/h = (E2 - E1)/h$$

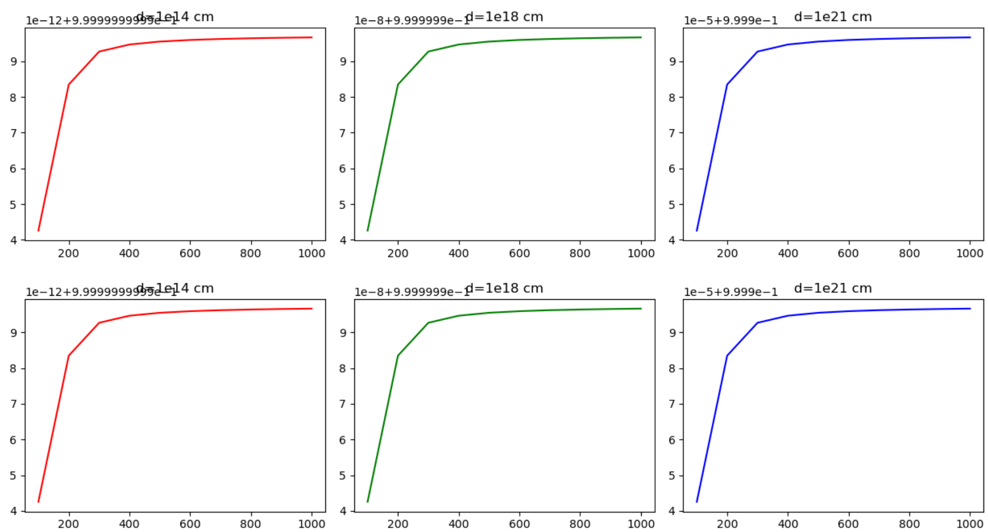
Note: The predictions are made based on the classifier as defined in the ipynb file.

Predicting a single prediction

```
[ ] le.inverse_transform(classifier.predict([[250,1340]]))

    array(['C II'], dtype=object)
```

Task 2: Astrophysical Absorption line exercise



This is the plot of **Intensity vs Wavelength(in nm)** of a light beam passing through a glass slab of varying thickness d. Here the first 3 figures represent the plots for $d = 10^{14}$, $d = 10^{18}$, $d = 10^{21}\ cm\ respectively$. The second row of figures represents the plots for the former values of d but with $\nu 0 = 2.46632 \times 10^{15} Hz$.

After analyzing all the 6 plots we infer that:

1. At initial wavelengths, the plot rises steeply for small changes of wavelength but after a certain threshold, it increases at very small rates. For example for $d = 10^{14} cm$ we see that the plot rises steeply until 300nm and after it remains almost constant. Similarly, for $d = 10^{18} cm$ it rises until 294nm and remains constant after that.
2. Even after changing $\nu 0 = 2.46632 \times 10^{15} Hz$ we see that there is a slight change in the plot. For example, for $d = 10^{14} cm$ At 400nm we find the intensity of the former is around *0.99999999999994655386359454496414400637149810791015625* while the latter has an intensity of *0.999999999999946564965824791215709410607814788818359375.*

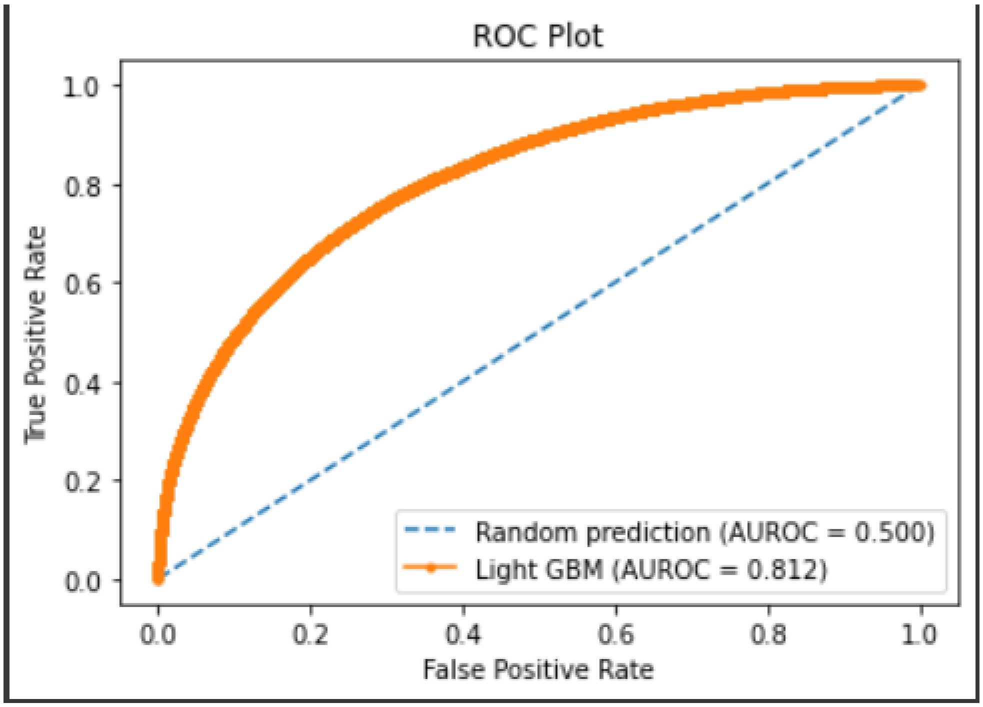
Hence, we conclude that changing the thickness of the slab significantly affects the plot while changing the value of *v0* did not significantly affect the plot.

Note: The plot is generated using matplotlib pyplot in python in the ipynb file.

Task 3: Dimensionality Reduction Exercise

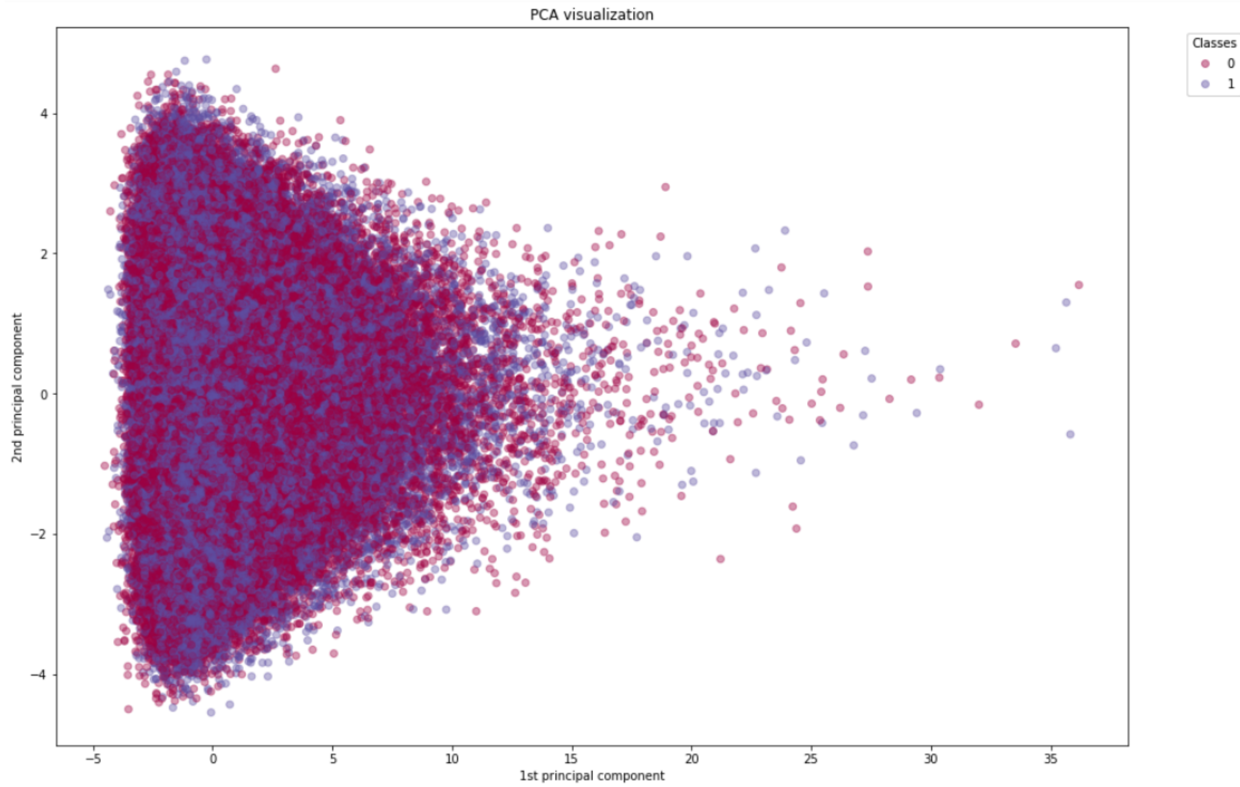
1. Using Principal Component Analysis (PCA)

A. ROC Curve for LIGHT GBM (Gradient Boosting Machine) Classifier before applying Dimensionality Reduction

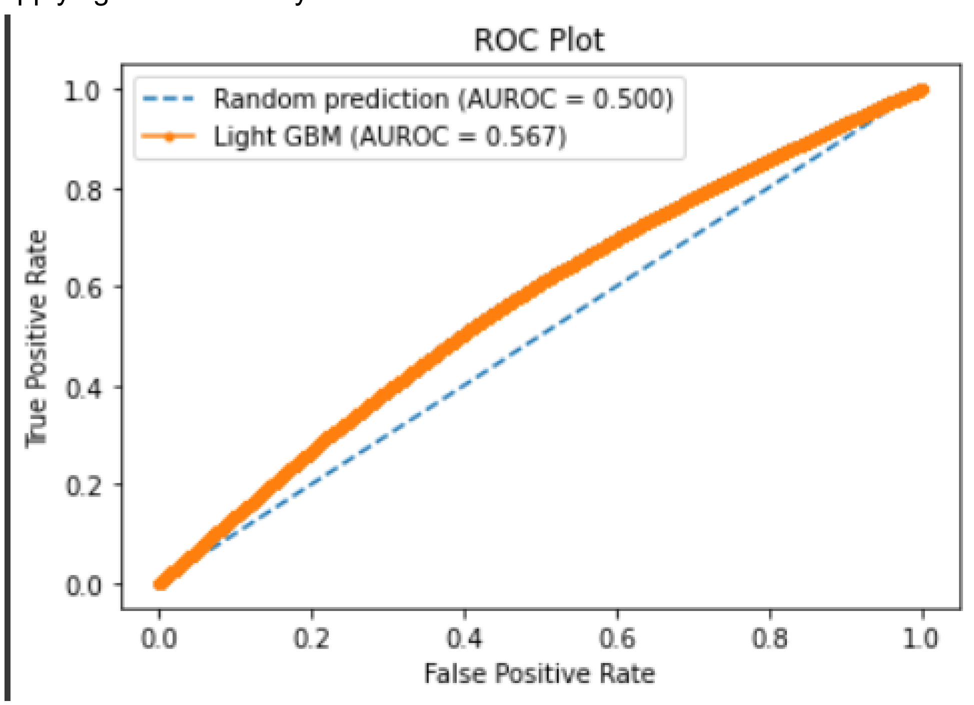


B.

- Visualizing the dataset after applying PCA



- ROC Curve for LIGHT GBM (Gradient Boosting Machine) Classifier after applying Dimensionality Reduction



The variance of each feature used after applying Principal Component Analysis(PCA)

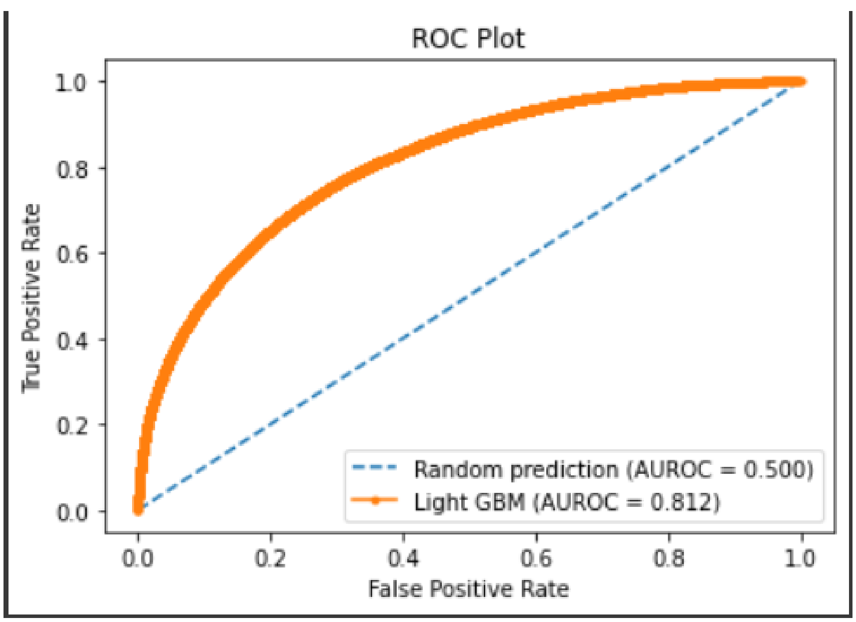
Printing the variance of each features selected by PCA (higher magnitude - higher importance)

```
[7] pca.explained_variance_ratio_

    array([0.14907311, 0.06656738])
```

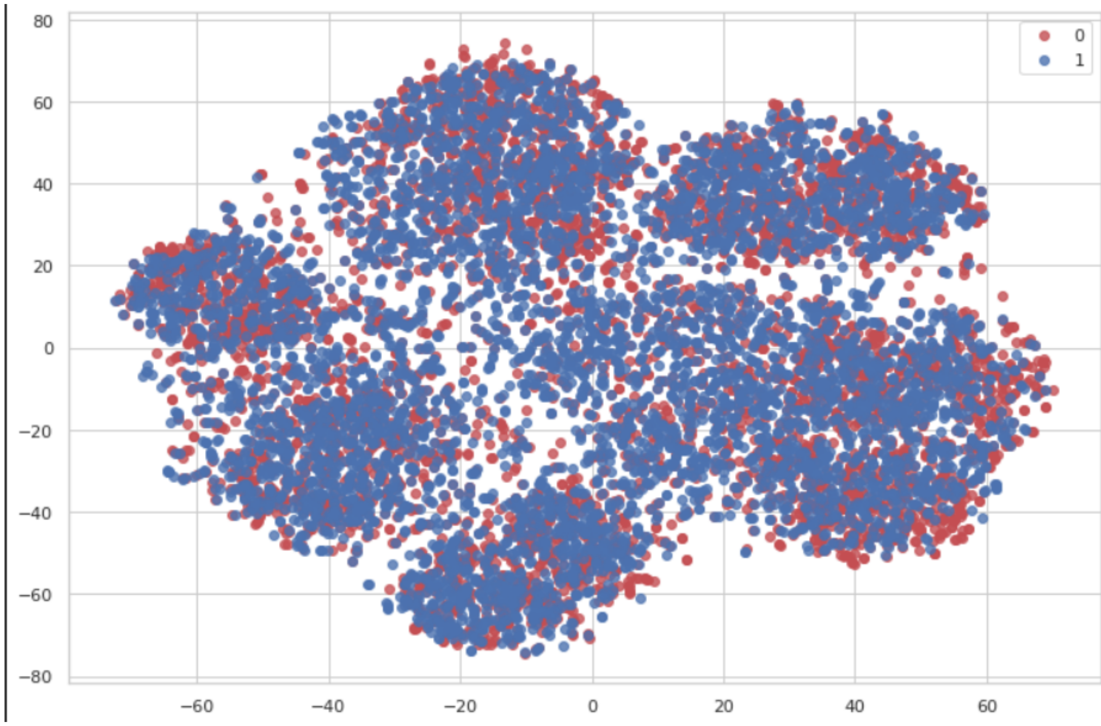
2. Using Autoencoders

A. ROC Curve for LIGHT GBM (Gradient Boosting Machine) Classifier before applying Dimensionality Reduction

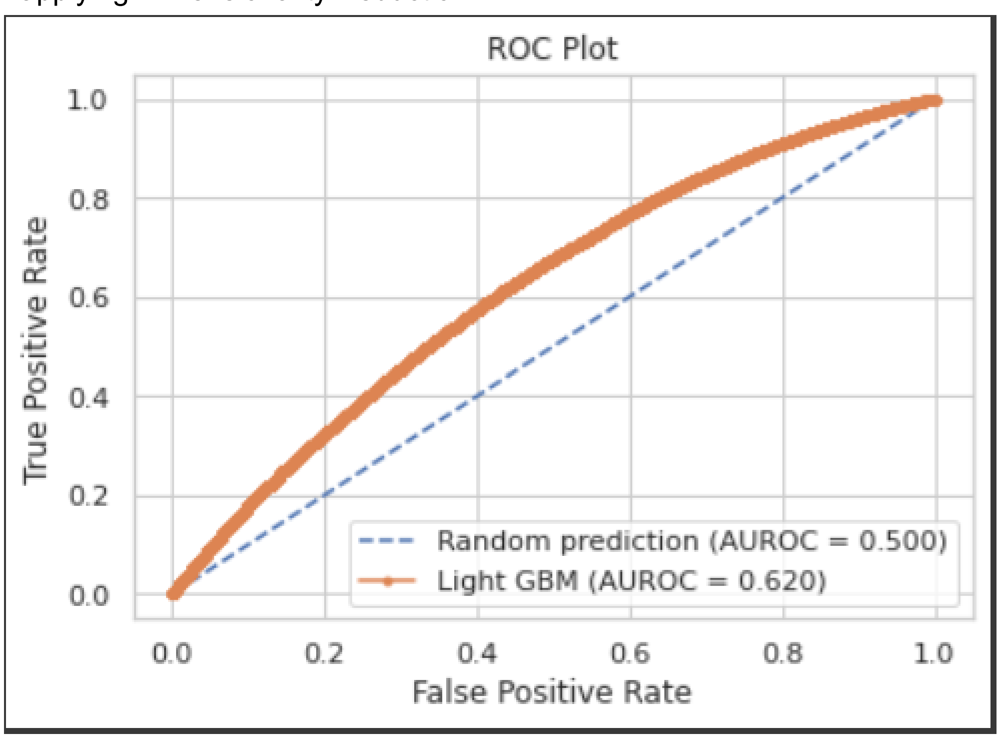


B.

- Visualizing the last 10k data using t-SNE



- ROC Curve for LIGHT GBM (Gradient Boosting Machine) Classifier after applying Dimensionality Reduction



The neural network built using 7 layers of encoders and decoders and tested using different activation functions for best results:

```
# Building the Input Layer
input_layer = Input(shape =(X.shape[1], ))

# Building the Encoder network
encoded = Dense(25, activation ='tanh',
                activity_regularizer = regularizers.l1(10e-5))(input_layer)
encoded = Dense(22, activation ='tanh',
                activity_regularizer = regularizers.l1(10e-5))(encoded)
encoded = Dense(19, activation ='tanh',
                activity_regularizer = regularizers.l1(10e-5))(encoded)
encoded = Dense(16, activation ='tanh',
                activity_regularizer = regularizers.l1(10e-5))(encoded)
encoded = Dense(13, activation ='tanh',
                activity_regularizer = regularizers.l1(10e-5))(encoded)
encoded = Dense(10, activation ='tanh',
                activity_regularizer = regularizers.l1(10e-5))(encoded)
encoded = Dense(7, activation ='tanh',
                activity_regularizer = regularizers.l1(10e-5))(encoded)

encoded = Dense(5, activation ='relu')(encoded)   ## latent space

# Building the Decoder network
decoded = Dense(7, activation ='relu')(encoded)
decoded = Dense(10, activation ='relu')(decoded)
decoded = Dense(13, activation ='relu')(decoded)
decoded = Dense(16, activation ='relu')(decoded)
decoded = Dense(19, activation ='relu')(decoded)
decoded = Dense(22, activation ='relu')(decoded)
decoded = Dense(25, activation ='relu')(decoded)

# Building the Output Layer
output_layer = Dense(X.shape[1], activation ='relu')(decoded)
```