# Cancer Subtype Classification from Gene Expression Data

**\*What does it mean?**

Cancer subtype classification based on gene expression data simply means taking cancerous samples, measuring the activity levels of thousands of their genes, and then using these unique 'gene activity fingerprints' to group similar tumours into distinct categories (subtypes). This helps us understand why different tumours of the same cancer type behave differently and how to best treat them.

The gene expression of each of the tumor affects the samples behaviour which especially include the ability to metabolize sugars to form ATP as a common pattern of overexpression of such genes has been observed in majority of the cancer samples followed with other behavioral patterns such as cell invasion, masking from immune cells (act as healthy cells or body’s own cells to remain undetected) and mobility.

**GOAL**

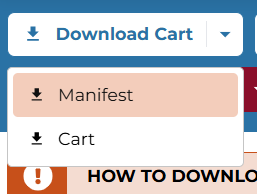
To build, evaluate, and deploy a machine learning model (XGBoost/ Random Forest) for classifying cancer subtypes from gene expression data, with strong emphasis on interpretability and user-friendly interaction.

**TOOLS**

Python, Pandas, NumPy, Scikit-learn, XGBoost/ Random Forest, UMAP, Matplotlib, Seaborn, Plotly, SHAP, Streamlit.

**STEPS FOLLOWED:**

**1. Identifying Suitable Dataset:**

* Publicly available, government approved, high-dimensional dataset for bladder cancer was downloaded from TCGA (<https://portal.gdc.cancer.gov/>).
* Gene expression profiles were required so, appropriate filters were used to do so. The process initiated with navigating to TCGA’s official bladder cancer database (<https://portal.gdc.cancer.gov/analysis_page?app=>) and a gene expression specific cohort was built by navigating to “[Cohort Builder](https://portal.gdc.cancer.gov/analysis_page?app=CohortBuilder&tab=general)” option and selecting “TCGA-BLCA” in the “Project” section.
* The above step helped to filter out bladder cancer specific samples, now to filter out gene expression profile “[Repository](https://portal.gdc.cancer.gov/analysis_page?app=Downloads)” feature was used.
* The quest to filter out necessary data which showcased only the gene expression profile was started by applying the following filters:
  + 1. Data Category: Transcriptome Profiling
    2. Data Type- Gene Expression Quantification
    3. Experimental Strategy- RNA-seq
    4. WorkFlow Type: STAR-Counts
    5. **The following hyperlink can be used to navigate to the website with the added filters: [CLICK HERE](https://portal.gdc.cancer.gov/analysis_page?app=Downloads)**
* All the files were added to the cart which helped to download them onto our system. The hyperlink helps to showcase how the cart looks like: [CART](https://portal.gdc.cancer.gov/cart)
* The manifest file was downloaded  , this file is a system-readable text file which contains individual download paths to our gene expression profiles.
* In order to download the gene expression profiles from the manifest file a dedicated transfer tool named “[GDC Data transfer tool](https://gdc.cancer.gov/access-data/gdc-data-transfer-tool)” was installed on the system. The download process was initiated after the following code was initiated in “Terminal” of the system:

**Code:** ./gdc-client.exe download -m gdc\_manifest.2025-06-08.013108.txt

**Note:** manifest.2025-06-08.013108.txt is the name of the manifest file.

* Observations and Outcome of this process:

1. Overall, **853** cases have been recorded in regard to bladder cancer in the NCI’s GDC data portal and 39380 files were there as of 7th June 2025.
2. There were total of 0.91% files (~412) present for the TCGA-BLCA (Bladder Urothelial Carcinoma).
3. Primary Site- Bladder
4. Tissue or Organ of origin- Anterior wall of Bladder, Bladder neck, Bladder, dome of bladder, lateral wall of bladder, overlapping lesion of bladder, posterior wall of bladder, Trigone of Bladder, Urachus, Ureteric Orifice.
5. After applying the respective filters, a total of 406 cases of Bladder cancer were shortlisted which contributed to 431 files.

**2. Data Processing**

The files which were downloaded exhibited an extension “.rna\_seq.augmented\_star\_gene\_counts.tsv” which contained several columns describing the sample but for the sake of this project only the “**Sample ID, Gene\_name, Gene\_type, Gene\_id, fpkm\_uq\_unstranded**” were needed and selected which accounted for gene data.

**The Code to do so have been provided in the “Codes for 1.ipynb (1.)”**

So, a file named TCGA\_BLCA.csv was obtained with gene expression profiles which are the numeric values stored in fpkm\_uq\_unstranded column.

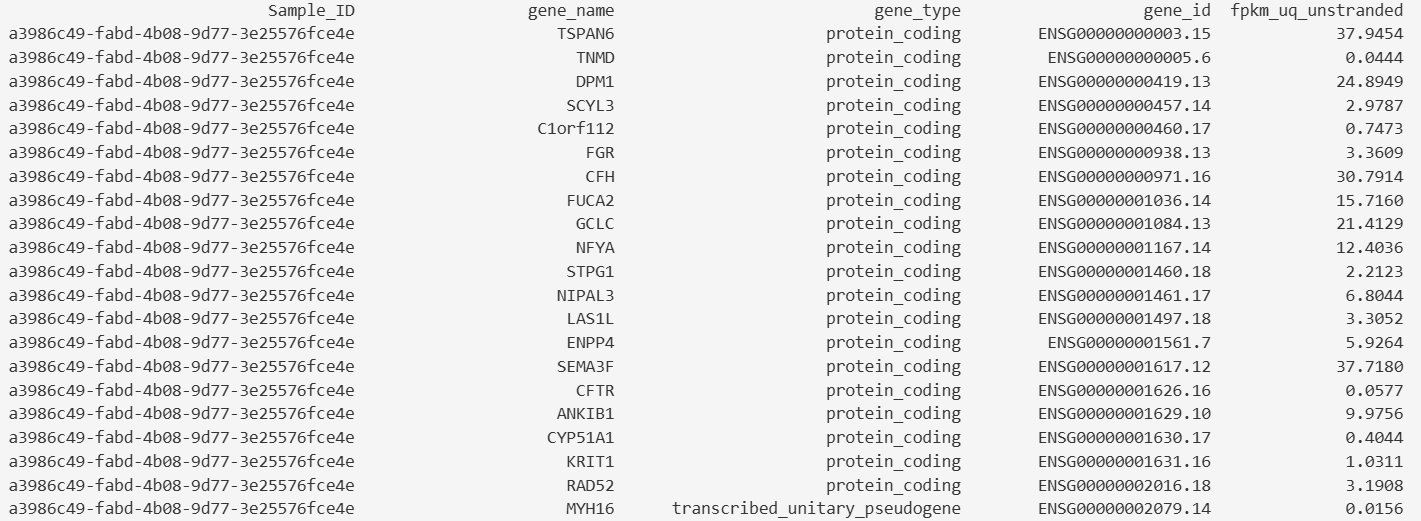
The Sample\_ID column contained names of the file (with the gene data) with their extension “.rna\_seq…” removed.

**Dimensions of the File:**

Rows: 25841160

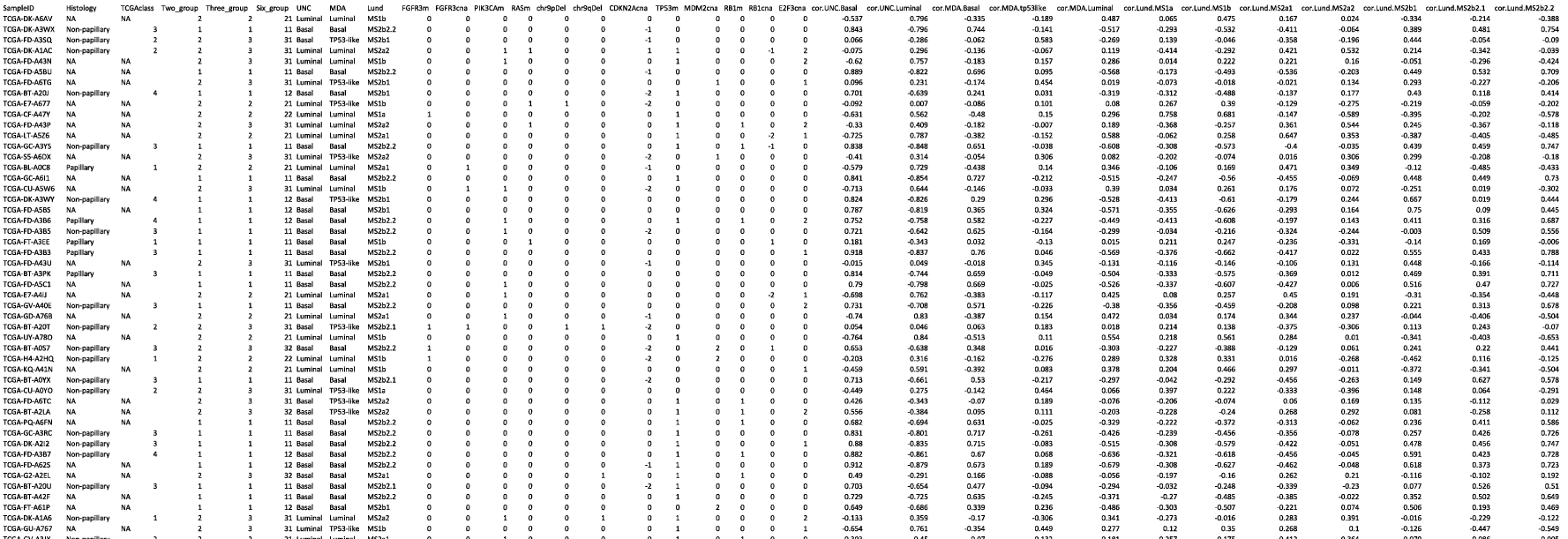
Columns: 5

**Output: For 1 sample.**

****

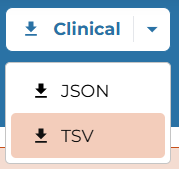
**3. Harvesting Labels:**

A thorough search was conducted on the webspaces to find out labels for all the samples which TCGA’s database dealt with wherein we found a research paper published by TCGA itself which contained a table with samples and their labels. The image below showcases the table:



The table contained sample IDs mapped according to TCGA’s ID so, we needed to map the appropriate samples with their gene expression profiles.

**4. Sample Mapping:**

* As discussed above our “TCGA-BLCA.csv” file contained SampleIDs which were according to the name of the file but our labels were according to the naming convention as per their guidelines. So, we downloaded a “[Clinical.tsv](https://portal.gdc.cancer.gov/cart)” file  was downloaded and converted to CSV format for better processing.
* Another metadata file  was downloaded from the website and later converted to CSV format.
* **Mapping Process:**

A. The aim of this process was to map the labels provided in the research paper-based table with their respective gene expression profiles.

B. “Clinical” file contained sample names which matched with the sample names present in the labels table. These sample names were used as a common mapping point between both the tables, meaning a new table named “Arranged\_Table.csv” was created using python code which filled the common matching sample name as the first column and their labels. Now, each sample present in the Clinical file contained unique IDs these were filled as well.

C. The Unique IDs were now the common ground between the metadata and Clinical files. Meaning the IDs were verified and if they matched appropriately then file names for that particular samples were filled in the new arranged table.

D. These file names matched with the file names present in our initially generated “TCGA-BLCA.csv” file which were then used as a common mapping parameter to fill out the gene data containing the gene expression profile.

**The 2a. – 2c. in “Codes for 1.ipynb” denotes the entire process.**

**5. Data Refining:**

The final output file contained zero-data columns or “Unnamed:x” where x was from an integer, these columns were created by the code itself to fill out the unwanted spaces which were generated when conversions to CSV were done.

**The 3 in “Codes for 1.ipynb” denotes the entire process.**

**Data Dimensions after processing: Rows:** 13891140

**Columns:** 40

**6. Applying PCA and UMAP for dimensionality Reduction:**

* As stated earlier, the processed data is high dimensional so, it needs to be reduced for PCA and UMAP to be deployed on them.
* One way out of many is to convert any high dimensional data into a wide format meaning our data in its wide format has samples as rows and individual genes as columns with their gene expression profiles. This is made possible because each sample’s gene expression profile has been recorded for identical genes by the organization.

**“#4” Represents the step to convert the dataframe into a wide format.**

**Dimension of this data:**

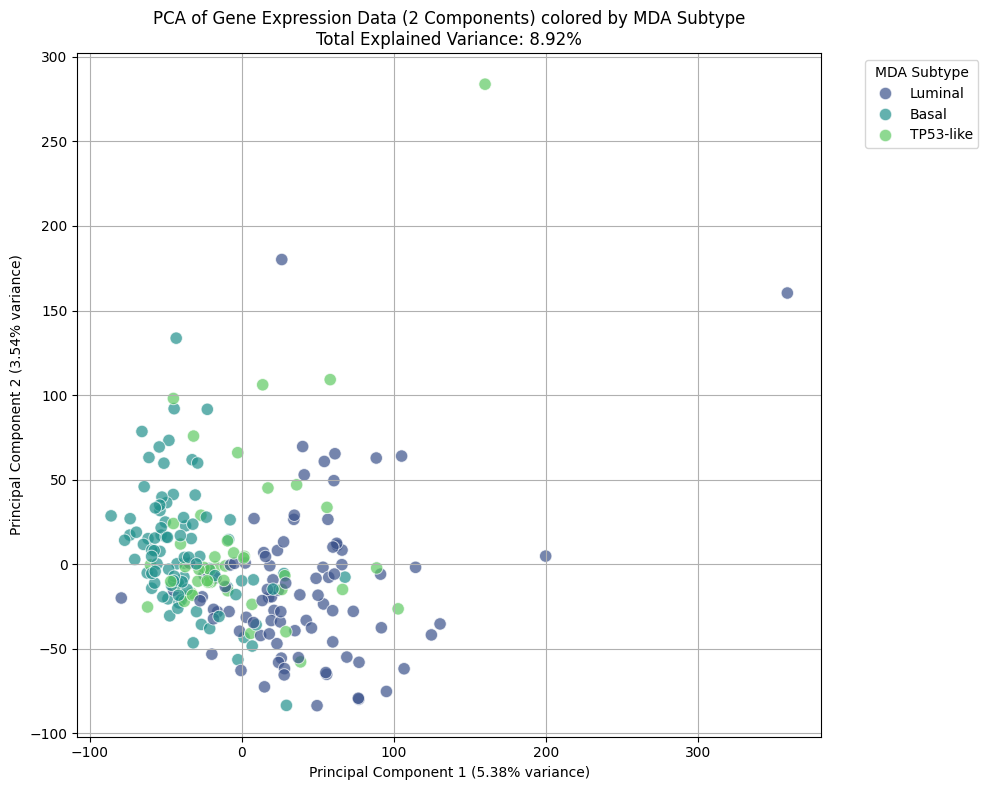
**Rows:** 229

**Columns:** 59427

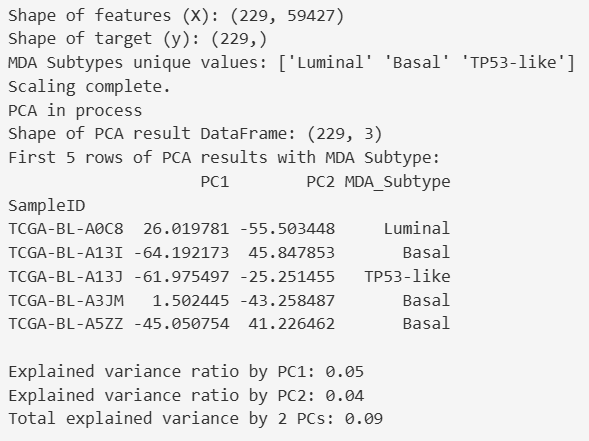
**7. PCA: Principal Component Analysis**

The process captures patterns based on significance between data points of the dataset, generally removes redundancy and improves the overall model performance thereby revealing hidden patterns.

The PCA-Ready data has been subjected to PCA techniques using appropriate modules like sklearn.preprocessing which normalizes the data meaning it revolves our data around 0 with a natural variance of 1. Using sklearn.decomposition to import PCA and visualization with seaborn helped to create the following plot:



**Code for the process is “#5”.**

****

**MDA subtypes Distribution (Using Count function):**Luminal- 92 (Meaning 92 bladder cancer samples classified as Luminal)

Basal- 90 (Meaning 90 bladder cancer samples classified as Basal)

TP53-like- 47 (Meaning 47 bladder cancer samples classified as TP52-like)

The PCA-Ready data was further scaled using StandardScaler as mentioned it being extremely case sensitive (sensitive to minute variances and scaling helps to increase sensitivity).

Shape of New dataframe: 229 Rows

Columns 3 (**PC1, PC2, MDA-Labels**).

Graph:

The graph denotes an explained variance of 8.92% now, variance here means the information being covered or simply how much information is being covered by the current dimensional data or by the PCA technique. Higher variance is generally preferred as it denotes much more variance and depicts inclusion of more data or information thereby gives a definite idea on the structure of the original data. This variability holds the information that differentiates our samples.

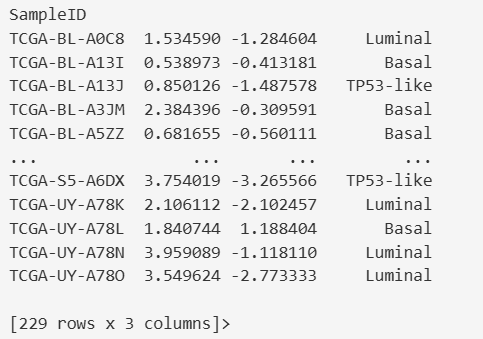
PC1 captures the single most axis of variability meaning it captures the highest variance out of our reduced data. It identifies the most significant way the samples differ based on the entire gene expression profiles. PC2 is the second most axis of variability meaning it captures the next most significant way the samples differ based on their expression profiles.

There were labels provided by UNC, Lund University and MDA out of them we have selected labels given for each sample by MDA only.

**8. UMAP: Uniform Manifold Approximation Projection**

This is a dimensionality reduction technique but unlike PCA this preserves the local and global structure of the original high dimensional data which when plotted appears to be more clustered, distinct in organization. This is a non-linear dimensionality reduction technique which separates or plots points on the graph based on their difference or how far away they are from each other.

It establishes a relationship between data points (Instead of variance like done in PCA techniques) and gives an idea on how similar or dissimilar the data points are in the high dimensional space. So, naturally points that are closer together in high-dimensional space (In terms of their expression values) would appear closer together on the plot whereas those which are far away to each other (In terms of values) appear farther to each other on the plot.

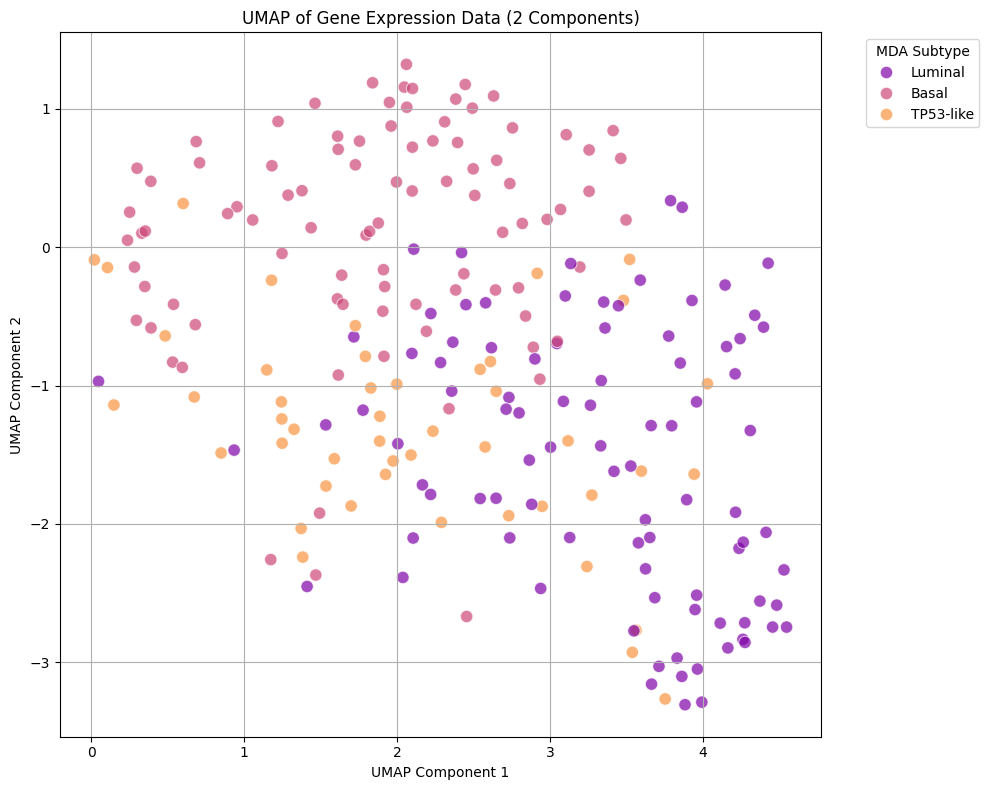


UMAP\_1 represents the X-coordinate of Sample in the Plot.

UMAP\_2 represents the Y-coordinate of Sample in the Plot.

Looking at these coordinates helps us which gene expression profile of which sample is closer with that of the other (like X (1.5, 1.6) is closer when Y (-0.5,-0.6)). The coordinates give an idea of the structure of gene expression profile of that particular sample in the original high-dimensional data.

The graph below visualizes the same more comprehensively:

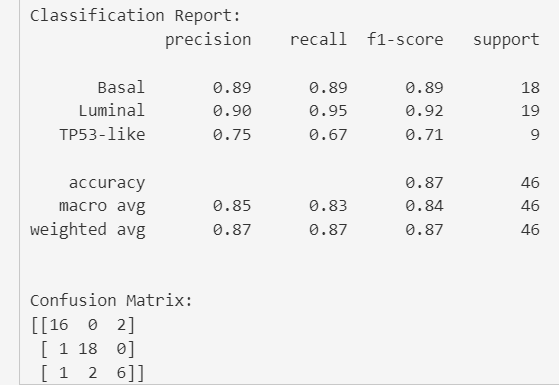


**“#6” denotes the UMAP’s code.**

**9. Training Random Forest**

This is an algorithmic based classifier which does not overfit and uses ensemble of decision trees to make predictions.





When the decimal is multiplied by 100 (x100) it denotes the percentage of accuracy classified by Random Forest module. So, overall accuracy of approximately 87% (0.8696) was obtained which is considered to be very good based for extremely high-dimensional data especially gene expression profiles.

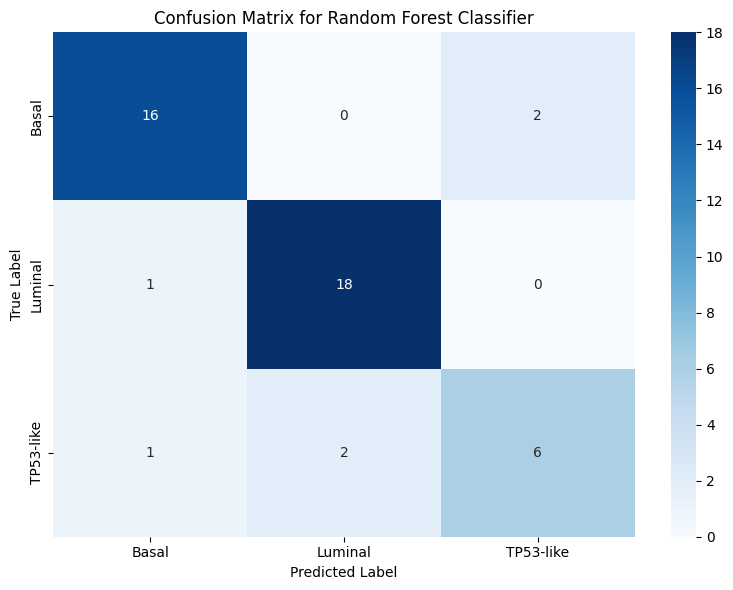
Claudin-low and Neuroendocrine labels have been removed because extremely low data was recorded or available in the dataset (both processed and original one).

Significance:

F1-score: Harmonic Mean of precision and recall. Recall is the model’s ability to correctly identify all relevant positive instances in the dataset or simply denotes how many samples were correctly identified.

Support denotes the occurrence of the labels in the dataset where low Support levels (<Support:5) denote extremely rare occurrences which was found or evident in Neuroendocrine and Claudin-low label’s cases.

Support 18 (In case of Basal) means that there were 18 actual basal labelled samples in our processed dataset (test data), keep in mind that the dataset was split in an 80 to 20 ratio meaning 80% was the training data (model would be trained on this) and rest 20% is the test data so, 16 coming out to be truly-labelled increases the accuracy of the model

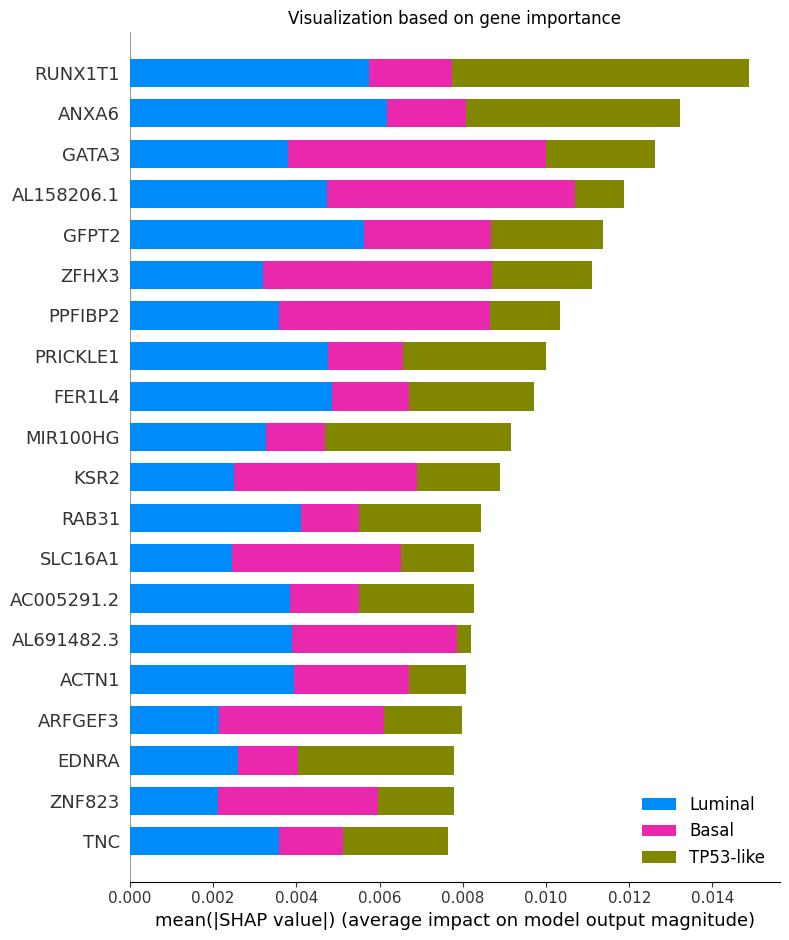


**“#7” denotes the code for the same process.**

**10. Visualizing gene importance using SHAP: Shapley Additive Explanations**

Method which explains the output of any machine learning model by providing a way to understand how each input contributes and, in this case, this predicts how each individual expression profile of the gene contributes to model’s prediction which gives an accurate idea about its importance. So, simply the more the gene contributes to the model’s prediction the more importance it holds.

The retraining of Random Forest ensures code robustness.



From this data RUNX1T1 gene appears to be the most impactful out of the other or simply to say the least, it affects the model’s prediction the most.

The overall plot denotes the most impactful and important genes chosen by our Random Forest model to predict and classify bladder cancer into sub-types.

The importance or impact of the gene on the classifying bladder cancer into subtypes decreases vertically meaning the gene at the top (RUNX1T1) indicates highest impact or importance which is contradictory to the gene at the bottom (TNC).

**“#8” denotes the code used for the visualization process.**

**11. Deploying the model on Streamlit:**

Streamlit is a web-based dashboard which allows to deploy machine learning based models without any prior knowledge of front-end or back-end development.

**A. Saving Resources**

* To ensure successful deployment, streamlit needs to be installed in VS code so, by using “pip install streamlit” and importing streamlit in the workspace we can now easily access the streamlit servers using our system’s Terminal.
* Prior to accessing the servers, streamlit needs to access the model and its required cache resources on the system. So, a dedicated directory has been created with all the required cache resources and the trained model being saved using joblib. Now, joblib is a package used to save models or resources in a “.joblib” format.
* Just to summarize the above description, we have installed streamlit in our system, then installed joblib, saved all the model and streamlit necessary resources in a dedicated directory called “trained\_models” in “.joblib” file format to allow streamlit to access the model and the resources after running it from our systems command line interface or Terminal.

**“#9” is the complete code to save all the necessary resources required for streamlit and model deployment.**

**B. Streamlit UI:**

As stated earlier, streamlit deployment on the web does not require prior knowledge of front-end or back-end development which makes designing a web-based application extremely easy when streamlit’s dedicated design-specific code lines are used.

For this project a simple but elegant web page has been developed with appropriate title, appropriate instructions on the format of the dataframe accepted all while maintaining the model’s deployment and its trained accuracy.

**“#10” covers the entire code of designing the UI**

**12. App.py**

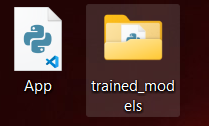
Now, streamlit requires a dedicated python script with the UI setup, directory’s path to load our streamlit resources and ability to run through the Command line interface or Terminal of our system. To cater all these requirements a dedicated python script has been coded in order to satisfy streamlined deployment and saved under the name of “App.py”

**Opening “App.py” with VS Code would help to read the code.**

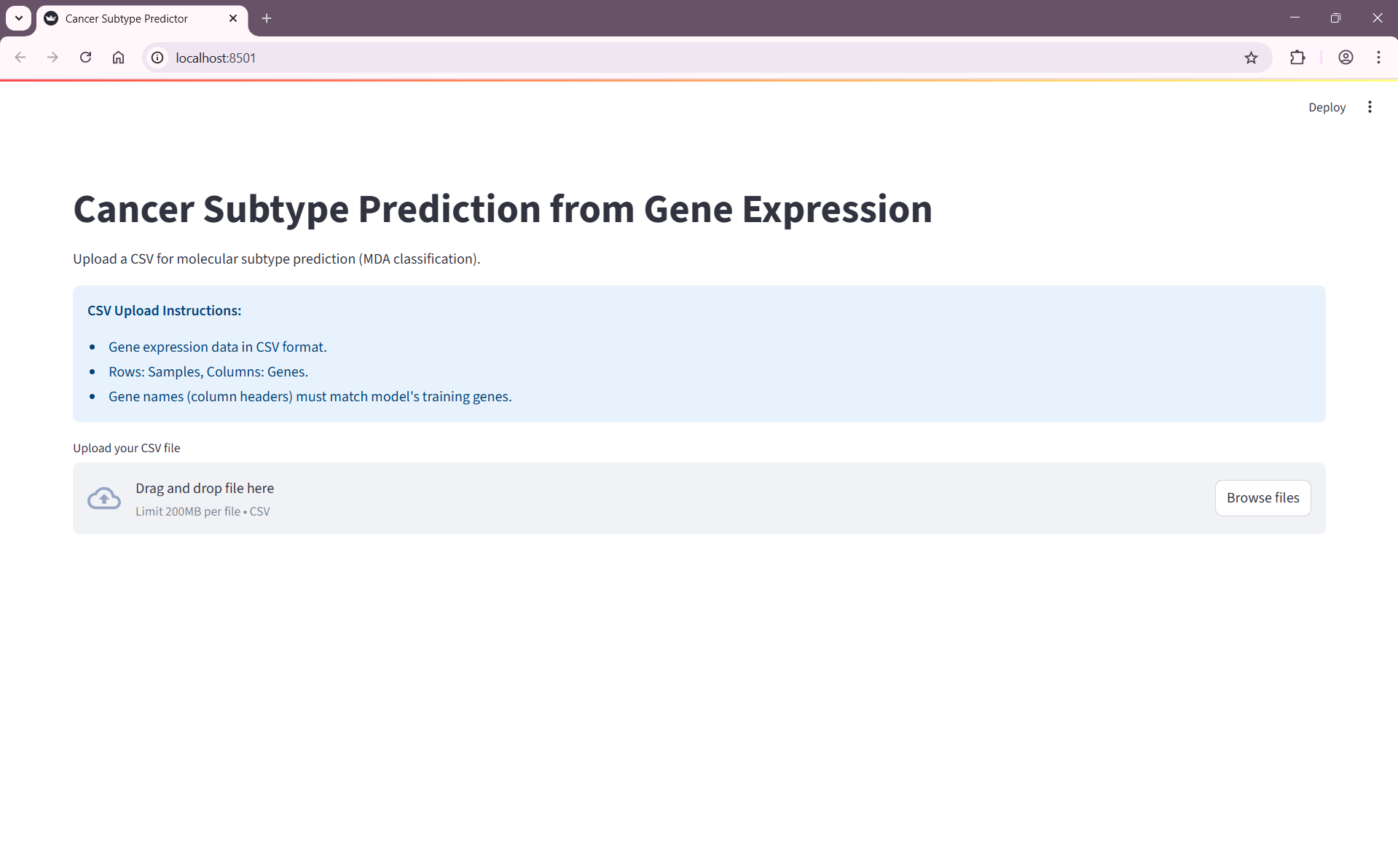
Now, the model can be deployed on a novel data for cancer subtype classification using Command Line Interface or Terminal by running the following lines of code:



To ensure easy processing, all the files in streamlit’s resource directory along with App.py have been moved to the Desktop.



The image below depicts the webpage:

**\*CONCLUSION:**

1. The above project classifies bladder cancer into its subtypes and after appropriate data processing, model training and testing the ultimate aim has been achieved.
2. The most prominent labels for bladder cancer have been found out to be Basal, Luminal and TP53-like.
3. The gene importance has been visualized which gives a comprehensive understanding on the most expressive genes of the samples and such visualization helps in improved and directed treatments.
4. Any novel data which follows the format instructions (which have been mentioned) can be exported to our streamlit server and classification can be done with prediction accuracy of 87% .