COMP3212: Computational Biology

Project 3:
Cancer Genome Study

Georgi Iliev

Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton
Email: gdi1u21@soton.ac.uk

May 2024

1. Overview

This study explores the UCSC Xena cancer genome database [1]. Specifically, it conducts experiments on the Genomic Data Commons (GDC) The Cancer Genome Atlas (TCGA) datasets for various cancers - Bladder, Breast, Cervical, Colon, Endometrioid, Esophagael and Glioblastoma using multiple statistical and machine learning algorithms. On one hand, the statistical data manipulation includes K-Means Clustering, Principal Component Analysis (PCA), Hierarchical Clustering, DBSCAN, Support Vector Machines (SVM), and evaluations using Silhouette Scores and Receiver Operating Characteristic (ROC) curves. This provides understanding of how data is spread, along with invaluable insights into the dependencies and relationships between the data. On the other hand, Machine Learning, including Convolutional Neural Networks (CNN), Random Forest Classifier and Logistic Regression with L1 Regularisation all support pattern detection and identification of important genes related to Breast Cancer.

2. K-Means Clustering and Silhouette Anakysis

Applying K-Means Clustering, which is a centroid-based clustering algorithm, allows partitioning of datasets into k clusters by minimising the sum of squared distance in each cluster [2]. Statistical determination of dependencies reduces the randomness of the data by ensuring their conservation, which leads to avoiding making assumptions about the initial centres [3].

K-Means clustering identifies groups based on gene expressions of $A2M$ and $A2ML1$ variables due to their significance in cancer development, progression, and metastasis. Results are visualised in *Figure 1*.

Figure 1: *K-Means Clustering of combined cancer datasets*

When executed independently on each cancer dataset, the algorithm was most effective in distinguishing between different stages of tumour progression in Bladder Urothelial Carcinoma (BLCA), as evidenced by clear cluster separations, illustrated in *Figure 2*, and high silhouette scores, indicating well-defined, compact clusters.

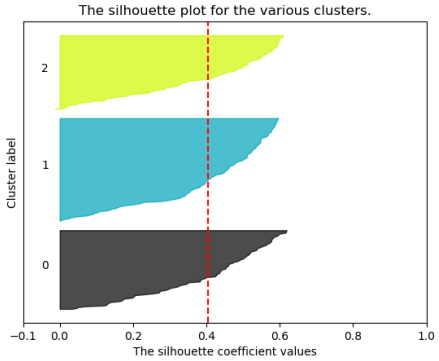

Figure 2: *Cluster Separations BLCA*

The results clearly show the association of these two gene expressions with tumour stages, making both K-Means and Silhouette Scores suitable for initial clustering and cancer analysis.

3. Principle Component Analysis (PCA)

Principal Components Analysis (PCA) is a statistical technique that simplifies the data complexity of original variables while retaining their underlying characteristics [4]. These newly obtained variables are projected into a lower-dimensional space and are called principal components (PCs). PCA effectively highlights key characteristics in a dataset and enhances the clarity of data clustering by showing the variance covered by the PCs [4].

PCA proved particularly effective in managing data from BRCA, UCEC, and GBM cancer studies, preserving substantial variance and facilitating clearer discrimination of data with less distinguishable prior transformation. The data distribution is illustrated in *Figure 3*, demonstrating how these three types of cancer are distinctly separated as opposed to other overlapping clusters.

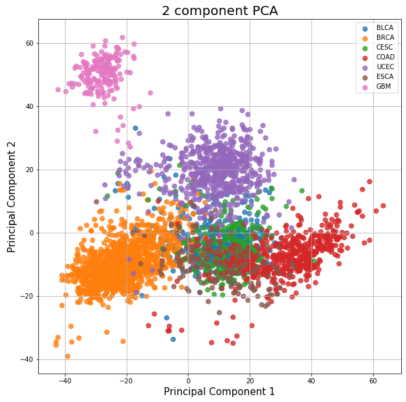

Figure 3: *PCA's data distribution among the first two PCs*

Principal components (PCs) and gene contributions are estimated after analysing the loadings for each PC. These

loadings are coefficients in the linear combinations forming the PCs and indicate the correlation between the original traits and the PCs, as described in the equation below.

$$Loadings = Eigenvectors \cdot \sqrt{Eigenvalues}$$

Positive loadings indicate correlation, while negative values indicate anti-correlation. The magnitude of the loading of a gene reflects its influence on the PC. The first four PCs showed higher correlations with the original genes, suggesting they more effectively represent the initial data.

Having access to the Gene Ontology Knowledgebase [5] allows the analysis of gene ontology (GO) enrichment analysis on explored cancer studies, with results summarised in Table 1. Although not all gene sets yield statistically significant results, the remaining data highlights the types of genes key to cancer differentiation in the dataset.

Principal Component	GO terms summary
1 (BLCA)	Mitosis, DNA Replication, Cell Cycle Checkpoints, Cellular Adhesion, Migration, Antigen Processing and Presentation, Antibody Immune Response, Keratinisation, Protein Oligomerization and Ubiquitination
2 (BRCA)	Mitosis, DNA Replication, Cell Cycle Checkpoints, Cellular Adhesion, Migration, Keratinisation, Neuron Differentiation, Adipogenesis, Angiogenesis, Glycolysis, Metabolism, Amino Acids Protein Oligomerization, Ubiquitination, Digestive System, Antigen Processing and Presentation, and Antibody Immune Response
3 (CESC)	Mitosis, DNA Replication, Cell Cycle Checkpoints, Bacterial Response, Antigen Processing and Presentation, Antibody Immune Response, Phagocytosis (Cellular Eating), Cellular Adhesion, Migration, and Keratinisation
4 (COAD)	Mitosis, DNA Replication, Cell Cycle Checkpoints, Glycolysis, Metabolism, Amino Acids, Protein Oligomerization, Ubiquitination, Digestive System, Cellular Adhesion, Migration, Antigen Processing and Presentation, and Antibody Immune Response
6 (UCEC)	Mitosis, DNA Replication, Cell Cycle Checkpoints, Cellular Adhesion, Migration, Keratinization, Neuron Differentiation, Adipogenesis, Angiogenesis, Glycolysis, Metabolism, Amino Acids, Protein Oligomerization, Ubiquitination, Digestive System, Bacterial Response, Antigen Processing and Presentation, Antibody Immune Response, and Phagocytosis (Cellular Eating)
7 (ESCA)	Bacterial Response, Antigen Processing and Presentation, Antibody Immune Response Phagocytosis (Cellular Eating), Cellular Adhesion and Migration, Keratinization, Mitosis, DNA Replication, Cell Cycle Checkpoints, Oxidative Burst, Neuron Differentiation, Adipogenesis, Angiogenesis, Glycolysis, Metabolism, and Amino Acids
8 (GBM)	Mitosis, DNA Replication, Cell Cycle Checkpoints, Cellular Adhesion, Migration, Neuron Differentiation, Adipogenesis, Angiogenesis, Glycolysis, Metabolism, Amino Acids, Protein Oligomerization, Ubiquitination, Digestive System, Bacterial Response, Antigen Processing and Presentation, and Antibody Immune Response Phagocytosis (Cellular Eating)

TABLE 1: Summary of GO terms for each principal component

4. Hierarchical Clustering

This method produces a dendrogram that reveals insights into the data structure and demonstrates the nesting and relationships of clusters, vital for recognising genetic similarities and variations in cancer profiles. The method is particularly effective with the UCEC (Uterine Corpus Endometrial Carcinoma) dataset, offering an in-depth view of molecular subtypes and identifying distinct clusters that closely align with established subtypes, as depicted in *Figure 4*.

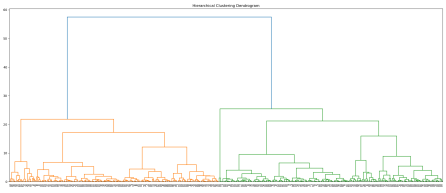

Figure 4: *Results from Hierarchical Clustering on the UCEC Cancer Dataset*

5. Density-Based Clustering Non-Parametric Algorithm (DBSCAN)

The DBSCAN determines a specific number of clusters based on the eps and minimum sampling parameters, which provides insights into the density of the explored datasets.

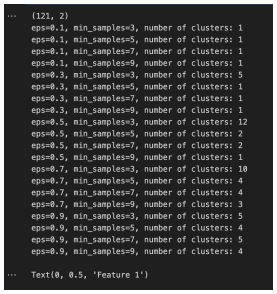

Figure 5: *Results from DBSCAN on the UCEC Cancer Dataset*

This technique was particularly effective in spotting outliers and core groups within the complex and noisy Esophageal Cancer (ESCA) dataset. The quantitative findings from this analysis are displayed in *Figure 5*. The robustness of DBSCAN in effectively handling noise makes it ideal for datasets where identifying outliers is critical to understanding disease mechanisms or treatment outliers.

6. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm applied to both classification and regression tasks. This project employs Support Vector Classification (SVC), a form of SVM used for classification. In SVC,

the model is trained with samples from various classes to predict the class of new samples. SVC operates as a non-probabilistic classifier, meaning it determines the most likely class for a given sample rather than providing a probability distribution across all classes [6]. In this study, a concrete linear kernel is used, and the optimisation problem is to minimise $\|\vec{w}\|$ while adhering to the constraints that define the decision boundary, formulated as follows:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \text{ for all } 1 \leq i \leq n[6]$$

Sklearn support vector functionalities were used for building a confusion matrix along with a classification report on the raw combined dataset. This was essential for understanding how SVM features analyse the data. The results are shown in *Figure 6* below.

```
...  [[609   0   0   0   0]
     [  0 173 370   0   0]
     [ 89   0   0   0   0]
     [  0 413   0   0   0]
     [270   0 453   0   0]]
                  precision    recall  f1-score   support

               0       0.63      1.00      0.77       609
               1       0.30      0.32      0.31       543
               2       0.00      0.00      0.00        89
               3       0.00      0.00      0.00       413
               4       0.00      0.00      0.00       723

        accuracy                           0.33      2377
       macro avg       0.18      0.26      0.22      2377
    weighted avg       0.23      0.33      0.27      2377
```

Figure 6: *Confusion Matrix and Classification Report executed using on the raw combined dataset*

6.1. OVO(One-vs-One) for Multi-Class Classification

SVMs are primarily binary classifiers. In order to adapt SVMs for multi-classification tasks, such as predicting different types of cancer, the "One vs. One" (OVO) strategy is implemented. This method constructs multiple classifiers by partitioning the dataset into binary classification tasks. The total number of classifiers is determined by the formula $(N * (N - 1))/2$, where N represents the total number of classes. For instance, given four classes — A, B, C, and D - OVO breaks down the task into:

- Classifier 1: A vs. B
- Classifier 2: A vs. C
- Classifier 3: A vs. D
- Classifier 4: B vs. C
- Classifier 5: B vs. D
- Classifier 6: C vs. D

6.2. ROC Curves Analysis

The receiver operating characteristic (ROC) curve is a graphical representation that demonstrates the performance of a classification model at various threshold levels by comparing the True Positive Rate against the False Positive Rate [7]. A ROC curve close to 1.0 indicates excellent model performance. Initially, this study conducts an experiment with inverted labels, leading to a performance that appears worse than random guessing, as shown in *Figure 7*.

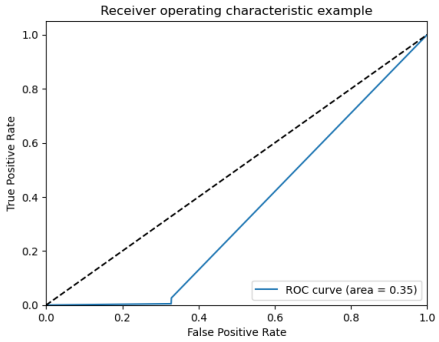

Figure 7: *ROC curve before inversion*

After a consultation with one of the lecturers who suggested inverting the labels, the issue was addressed, as depicted in *Figure 8*.

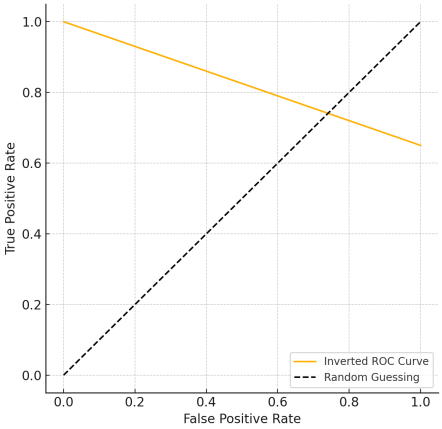

Figure 8: *ROC curve after inversion*

The revised ROC curve values indicate that the models performed effectively in distinguishing between different cancer types. This analysis was essential for evaluating the performance of a SVM classifier when applied to a dataset containing multiple subtypes.

7. Heatmap Analysis

A heatmap analysis is conducted to visualise gene expression data across various samples, focusing specifically on the A2M and A2ML1 genes and leading to a correlation with cancer types or severity, as shown in *Figure 9*. The A2M gene is well-known for its role in the immune response and inflammation [8]. It displays predominantly high levels of expression in most samples, marked by a consistent yellow coloring. This indicates active biological processes in the

sampled tissues or cells. As opposed, A2ML1 shows mostly low expression levels, highlighted by teal to purple colours, indicating less activity or different regulation compared to A2M in the same samples.

Figure 9: *Heatmap visualisation of A2M and A2ML1 genes across multiple samples*

8. Genetic Analysis

This study further explores key genes associated with breast cancer and aims to understand their functions using various computational biology techniques. By integrating somatic mutation data with patient survival information, a detailed genomic study of the BRCA dataset is performed. This leads to the formulation of a hypothesis regarding specific gene alterations that can potentially predict patient outcomes. These are also essential for understanding the molecular processes behind the disease.

8.1. Data Insights

The methodology begins with a rigorous preprocessing of merged datasets to create a clean, normalised framework for analysis. This includes removing any missing values, converting categorical variables to numeric format, standardising, and reshaping the data into the correct feature dimension format used in ML models. Such preprocessing provides valuable insights, such as the relationship between mutation effects and survival time in days, as illustrated in *Figure 10*.

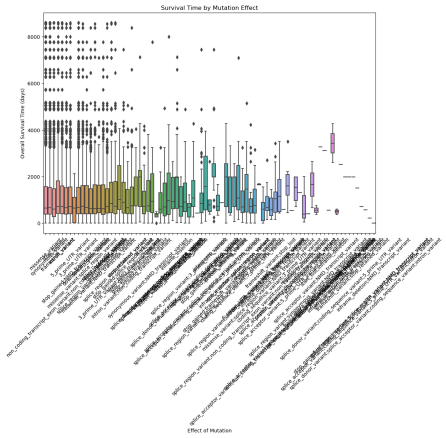

Figure 10: *Survival Time by Mutation Effect*

These observations allowed the identification of the most frequent mutations and their corresponding survival times, as illustrated in *Figure 11*. Upon comparison with reliable resources, it has been confirmed that these results are consistent with data from The Caner Genome Atlas Programme (TCGA).

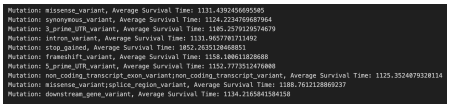

Figure 11: *Most Frequent Mutations with Average Survival Time*

8.2. Logistic Regression with L1 Regularization and Random Forest

Preliminary studies were conducted using logistic regression with L1 regularisation, effectively identifying several key genes, including ST6GALNAC3 and SYNPO2L. These genes are believed to play a role in breast cancer given their biological roles in processes such as cellular adhesion and immune response modulation. Among the genes analysed, ST6GALNAC3 was the most significant, showing an importance score of 0.723243. The remaining genes are detailed in *Figure 12*.

	Gene	Importance
15422	ST6GALNAC3	0.723243
12807	R3HDML	0.600900
309	ADA	0.450616
16576	TOM1	0.439789
15679	SYNPO2L	0.433539
10109	NCF4	0.427211
13933	SAMD10	0.426732
17236	UBXN6	0.413768
5667	FMNL3	0.395209
15449	STAP1	0.378791
11278	PACS2	0.377439
15483	STIM1	0.374968
15774	TAPBP	0.364577
6326	GPHA2	0.363444
161	AC092071.1	0.360435
10077	NBPF14	0.350815
17541	VSIG1	0.346044
12947	RAP1GDS1	0.344751
16345	TMEM190	0.340873
14183	SEPT14	0.337148

Figure 12: *Genes with most importance*

To further support the hypothesis and measure the influence of different genetic attributes, a Random Forest classifier was deployed. This approach highlighted a ranking of gene significance and verified the predictive capabilities of identified genes. The sequence of gene importance matched the rankings presented in *Figure 12*. thus validating our hypothesis.

8.3. Convolutional Neural Networks

The integration of a Convolutional Neural Network (CNN), equipped with an attention mechanism, offered two key advantages: precise outcome prediction and insights into which genetic regions are prioritised by model, both of which positively confirmed the initial hypothesis. The model's training regimen is depicted in *Figure 13*. It demonstrates high accuracy in both training and validation phases, with a split of 80% and 20%, respectively. Notably, while training losses show a significant reduction, validation losses do not decrease as expected,suggesting overfitting. Nevertheless, expanding the dataset for analysis, which could potentially address this problem, was considered too resource-intensive and not considered essential for this task.

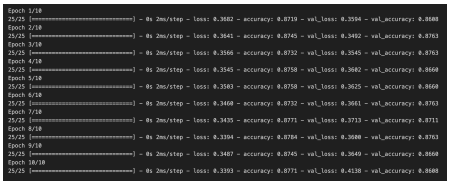

Figure 13: *Training regimen of CNN*

9. Conclusion and Future Work

In this comprehensive study, a diverse computational approach is used to analyse the complex landscape of cancer genomics using the UCSC Xena database and The Cancer Genome Atlas (TCGA) datasets. By applying advanced computational techniques such as K-Means Clustering, PCA, Hierarchical Clustering, DBSCAN, and SVM, significant insights into gene expression across various cancer types are obtained. This has improved the overall understanding of cancer biology.

9.1. Conclusion

Overall, the study not only achieved its aim of revealing unique expression patterns of various cancers and identifying key genetic markers linked to breast cancer but also advanced the methodology for analysing complex genomic data. By leveraging both statistical and ML computational biology techniques, the research has laid a foundation for future studies that could incorporate an even broader array of computational strategies, such as deep learning models for more granular analysis of gene expression data, potentially leading to groundbreaking discoveries in cancer genomics.

9.2. Future Work

- **Expansion of Gene Sets** Including a wider range of genes and using whole-genome sequencing data might reveal more biomarkers and provide a more complete picture of the cancer genomic landscape.
- **Advanced Machine Learning Models** Adopting more advanced machine learning techniques, such as Recurrent Neural Networks (RNNs), could improve the accuracy and forecasting power of classifying cancer types and forecasting outcomes.

As demonstrated throughout this and many other cancer research studies, the application of machine learning and data analysis techniques to cancer genomics can yield significant insights. Such insights can lead to the development of sophisticated AI models for predicting structures and interactions between molecules of different cancer types. One such recently developed model is the Google AlphaFold 3 [9], which was launched just a few days ago. The precision and capabilities of such models, together with their transformative nature in drug discovery, can improve the overall understanding of biological processes and lead to innovative treatments and therapies. For instance, AlphaFold's breakthroughs in protein folding offer promising ways to predict molecular structures and interactions across diverse cancer types. The application of these cutting-edge technologies in future research could accelerate the discovery of viable cancer treatment options and provide profound insights into the mechanisms of the disease.

References

[1] UCSC Xena, "UCSC Xena,"Accessed on: May 6, 2024.[Online].Available: https://xena.ucsc.edu/

[2] H. Xu et al.,"An unsupervised machine learning approach to evaluating the association of symptom clusters with adverse outcomes among older adults with advanced cancer: A secondary analysis of a randomized clinical trial,"*JAMA Netw. Open*, vol. 2, no. 3, Mar. 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10034574/

[3] Z. Kakushadze and W. Yu, "K-means and cluster models for cancer signatures," *Biomol. Detect. Quantif.*, vol. 12, pp. 15 24, Aug. 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5634820/

[4] F. Yao, J. Coquery, and K.-A. L. Cao, "Independent principal component analysis for biologically meaningful dimension reduction of large biological data sets," *BMC Bioinformatics*, vol. 13, no. 24, Feb. 2012. [Online]. Available:https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-24

[5] Gene Ontology Consortium, "Gene Ontology Resource," Accessed on: May 6, 2024. [Online]. Available: https://geneontology.org/

[6] I. Guyon et al., "Gene Selection for Cancer Classification Using Support Vector Machines," *Machine Learning*, vol.46, pp. 389–422, Jan. 2002. doi: 10.1023/A:1012487302797.

[7] Google, "Classification: ROC curve and AUC—Machine Learning—Google Developers," Accessed on: May 6, 2024. [Online]. Available: https://developers.google.com/machine learning/crash-course/classification/roc-and-auc

[8] K. Fang et al., "Screening of a novel upregulated lncRNA, A2M AS1, that promotes invasion and migration and signifies poor prognosis in breast cancer," *BioMed Res. Int.*, Apr.2020. [Online].Available:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7171613/

[9] team, G. D. A. (2024, May 8). Alphafold 3 predicts the structure and interactions of all of life's molecules. Google. https://blog.google/technology/ai/google-deepmind-isomorphic-alphafold-3-ai-model/life-molecules

10. Appendix

```
            sample    A2M  A2ML1   Dataset
0  TCGA-BT-A20W-11A  24.84  13.18  blca.tsv
1  TCGA-XF-AAN4-01A  24.69  16.02  blca.tsv
2  TCGA-SY-A9G0-01A  24.14   7.58  blca.tsv
3  TCGA-ZF-AA5P-01A  23.83  14.55  blca.tsv
4  TCGA-K4-A5RI-11A  23.79  10.98  blca.tsv
```

```
X = combined_data[['A2M', 'A2ML1']]
```

```
from sklearn.model_selection import train_test_split

# Split data into training and test sets
X_train, X_test = train_test_split(X, test_size=0.3, random_state=42)
```

```
from sklearn.cluster import KMeans

# KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_train)

# Visualizing clusters
import matplotlib.pyplot as plt

plt.scatter(X_train['A2M'], X_train['A2ML1'], c=clusters, cmap='viridis')
plt.xlabel('A2M Expression')
plt.ylabel('A2ML1 Expression')
plt.title('Cluster of Gene Expressions')
plt.colorbar()
plt.show()
```

Code used for generating Figure 1

```
from sklearn.metrics import silhouette_samples
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np

# Calculate the silhouette scores for each sample
sample_silhouette_values = silhouette_samples(X_train, clusters)
fig, ax = plt.subplots(1, 1)
y_lower = 10
n_clusters = len(np.unique(clusters))
for i in range(n_clusters):
    # Aggregate the silhouette scores for samples belonging to cluster i, and sort them
    ith_cluster_silhouette_values = sample_silhouette_values[clusters == i]
    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.nipy_spectral(float(i) / n_clusters)
    ax.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, facecolor=color, edgecolor=color, alpha=0.7)

    # Label the silhouette plots with their cluster numbers at the middle
    ax.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    # Compute the new y_lower for next plot
    y_lower = y_upper + 10

ax.set_title("The silhouette plot for the various clusters.")
ax.set_xlabel("The silhouette coefficient values")
ax.set_ylabel("Cluster label")

# The vertical line for average silhouette score of all the values
ax.axvline(x=silhouette_avg, color="red", linestyle="--")

ax.set_yticks([])  # Clear the yaxis labels / ticks
ax.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
plt.show()
```

Code used for generating Figure 2

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Separating out the features
features = ['A2M', 'A2ML1']
x = combined_data.loc[:, features].values
# Standardizing the features before applying PCA
x = StandardScaler().fit_transform(x)
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
# Create a DataFrame with the PCA results
principalDf = pd.DataFrame(data = principalComponents, columns =
                           ['principal component 1', 'principal component 2'])
# Concatenate the dataset labels to this DataFrame
finalDf = pd.concat([principalDf, combined_data[['Dataset']]], axis = 1)

# Plotting
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
# Color mapping by 'Dataset'
datasets = finalDf['Dataset'].unique()
colors = ['r', 'g', 'b', 'y', 'c', 'm', 'k']

for dataset, color in zip(datasets, colors):
    indicesToKeep = finalDf['Dataset'] == dataset
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(datasets)
ax.grid()
plt.show()
```

Code used for generating Figure 3

```python
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering

# Generate the linkage matrix
Z = linkage(X_train, 'ward')

# Plot the hierarchical clustering as a dendrogram.
plt.figure(figsize=(25, 10))
plt.title('Hierarchical Clustering Dendrogram')
dendrogram(Z)
plt.show()

# Apply Agglomerative Clustering
cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
cluster.fit_predict(X_train)
```

Code used for generating Figure 4

```python
from sklearn.cluster import DBSCAN
print(X_train.shape)

# Convert X_train to a numpy array
X_train_np = X_train.to_numpy()

# Try different values of eps and min_samples
for eps in [0.1, 0.3, 0.5, 0.7, 0.9]:
    for min_samples in [3, 5, 7, 9]:
        db = DBSCAN(eps=eps, min_samples=min_samples)
        db_clusters = db.fit_predict(X_train_np)
        print(f"eps={eps}, min_samples={min_samples}, number of clusters: {len(np.unique(db_clusters))}")

# If X_train has only one feature, plot it against the cluster assignments
if X_train_np.shape[1] == 1:
    plt.scatter(X_train_np[:, 0], db_clusters, cmap="plasma")
else:
    plt.scatter(X_train_np[:, 0], X_train_np[:, 1], c=db_clusters, cmap="plasma")
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```

Code used for generating Figure 5

```python
from sklearn import tree

# Train the model with the best parameters
clf = tree.DecisionTreeClassifier(**grid.best_params_)
clf.fit(X_train, clusters)

# Predict the labels
y_pred = clf.predict(X_train)

# Generate confusion matrix
print(confusion_matrix(clusters, y_pred))

# Generate classification report
print(classification_report(clusters, y_pred))
```

Code used for generating Figure 6

```python
# Predict the probabilities for Decision Tree
y_score = clf_tree.predict_proba(X_train)

# Compute ROC curve and ROC area for each class
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot of a ROC curve for a specific class
for i in range(n_classes):
    plt.figure()
    plt.plot(fpr[i], tpr[i], label='ROC curve (area = %0.2f)' % roc_auc[i])
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

Code used for generating Figure 7

```python
# Predict the probabilities for Decision Tree
y_score = clf_tree.predict_proba(X_train)

# Compute ROC curve and ROC area for each class
for i in range(n_classes):
    # Compute standard ROC curve
    fpr[i], tpr[i], _ = roc_curve(y[:, i], y_score[:, i])
    # Reverse the ROC by subtracting TPR and FPR from 1
    fpr[i] = 1 - fpr[i]
    tpr[i] = 1 - tpr[i]
    # Compute AUC on the reversed ROC Curve
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot of a reversed ROC curve for a specific class
for i in range(n_classes):
    plt.figure()
    plt.plot(fpr[i], tpr[i], label='Reversed ROC curve (area = %0.2f)' % roc_auc[i])
    plt.plot([0, 1], [1, 0], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Reversed Receiver operating characteristic example')
    plt.legend(loc="upper left")
    plt.show()
```

Code used for generating Figure 8

```python
import seaborn as sns

# Sort the data by cluster labels
X_train_sorted = X_train_clustered.sort_values(by='cluster')

# Drop the 'cluster' column for the heatmap
X_train_sorted_dropped = X_train_sorted.drop('cluster', axis=1)

# Create a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(X_train_sorted_dropped, cmap='viridis')
plt.title('Heatmap of gene expression data')
plt.show()
```

Code used for generating Figure 9

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram of DNA Variant Allele Frequency (VAF)
plt.figure(figsize=(10, 6))
sns.histplot(merged_data_cleaned['dna_vaf'], kde=True)
plt.title('Distribution of DNA Variant Allele Frequency')
plt.xlabel('DNA VAF')
plt.ylabel('Frequency')
plt.show()

# Boxplot for Overall Survival time grouped by gene mutation effect
plt.figure(figsize=(12, 8))
sns.boxplot(x='effect', y='OS.time', data=merged_data_cleaned)
plt.title('Survival Time by Mutation Effect')
plt.xlabel('Effect of Mutation')
plt.ylabel('Overall Survival Time (days)')
plt.xticks(rotation=45)
plt.show()
```

Code used for generating Figure 10

```python
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Fit logistic regression with L1 penalty
model = LogisticRegression(penalty='l1', solver='liblinear', random_state=42)
model.fit(X_train_scaled, y_train)

# Check coefficients
importances = pd.DataFrame(data={
    'Gene': genes_selected,
    'Importance': model.coef_[0]
})
importances = importances.sort_values(by='Importance', key=abs, ascending=False)
print(importances.head(20))
```

Code used for generating Figure 12