Module_3:

Team Members:

Julena Patel and Christy Lee

Project Title:

Utilizing Machine Learning to Predict Angiogenesis in Patients with Different Cancer Types

Project Goal:

The goal of this project is to determine the effect of cancer types of angiogenesis levels utilizing machine learning methods to develop a method of predicting angiogensis in patients with different cancer types. --> also looking at vegfc and vegfa

Disease Background:

Pick a hallmark to focus on, and figure out what genes you are interested in researching based on that decision. Then fill out the information below.

- Cancer hallmark focus:
  - Angiogenesis
- Overview of hallmark:
  - Angiogenesis is the sprouting of new blood vessels from old ones. This process typically occurs for just a short time to heal wounds, but in cancer cells, it can become persistent. The problem with angiogenesis is that it supports the expansion and progression of tumors. The resulting tumors often possess abnormal features, like extra branching, "leakiness," and microhemorrhaging.
- Genes associated with hallmark to be studied (describe the role of each gene, signaling pathway, or gene set you are going to investigate):
  - VEGF-A:
    - The VEGF-A protein is the main driver of angiogenesis. VEGF-A binds to and activates both VEGFR-1 and VEGFR-2, promoting angiogenesis, vascular permeability, cell migration, and gene expression.5 In addition, Lee et al. showed that an autocrine loop of VEGF-A and its receptor system exist within vascular endothelial cells, contributing to endothelial functions. VEGF-A has a feature called haploid insufficiency, meaning even one deficient gene will make the mutant embryo die early in the process. (https://pmc.ncbi.nlm.nih.gov/articles/PMC3411125/)
    - The VEGF-A signaling pathway promotes angiogenesis by binding to receptors such as FLT1, KDR, and NRP1 on endothelial cells, activating a kinase cascade involving RAS and MAPK. This cascade stimulates the growth of new blood vessels, which tumors can exploit to sustain their growth. Drugs like bevacizumab (a monoclonal antibody) and sorafenib and sunitinib (small molecule kinase inhibitors) disrupt this pathway by

blocking VEGF or its receptor activity, thereby inhibiting tumor angiogenesis. Variations in VEGFA and related genes can influence both cancer risk and patient response to these treatments. (https://www.clinpgx.org/pathway/PA2032)

Will you be focusing on a single cancer type or looking across cancer types? Depending on your decision, update this section to include relevant information about the disease at the appropriate level of detail. Regardless, each bullet point should be filled in. If you are looking at multiple cancer types, you should investigate differences between the types (e.g. what is the most prevalent cancer type? What type has the highest mortality rate?) and similarities (e.g. what sorts of treatments exist across the board for cancer patients? what is common to all cancers in terms of biological mechanisms?). Note that this is a smaller list than the initial 11 in Module 1.

- For now, we are only focusing on gliomas. We will choose other cancers to compare it to later but this is the primary type we want to learn about.
- Prevalence & incidence
  - Gliomas are the most common primary brain tumors of the central nervous system (CNS). In the United States, there are 6 cases of gliomas diagnosed per 100,000 individuals every year. There are an estimated 80,000 to 90,000 newly diagnosed cases of primary brain tumors each year in the United States, with approximately 25% being gliomas. The total number of glioblastoma cases diagnosed each year is about 12,000 (approximately 15% of the total of newly diagnosed brain tumors and roughly 50% of all malignant brain tumors). (https://www.ncbi.nlm.nih.gov/books/NBK441874/)
- Risk factors (genetic, lifestyle) & Societal determinants
  - History usually does not play a role in onset
  - However, some patients may have tumor-predisposition syndromes
  - patients with tuberous sclerosis are predisposed to having subependymal giant cell astrocytomas (https://www.ncbi.nlm.nih.gov/books/NBK441874/)
- Standard of care treatments (& reimbursement)
  - Surgical resection is the mainstay of treatment with a goal of maximal safe resection, depending on tumor grading and location.
    - Grade I is easiest and most likely to be cured, while grade IV requires specific surgeries, like stereotactic biopsy
  - Other treatments include chemoradiation, as well as antiepileptic medications, deep venous thrombosis prophylaxis, and steroids to help with symptoms (https://www.ncbi.nlm.nih.gov/books/NBK441874/)
- Biological mechanisms (anatomy, organ physiology, cell & molecular physiology)
  - Gliomas, particularly high-grade gliomas like glioblastoma, are aggressive brain tumors characterized by their ability to invade surrounding brain tissue. This invasion occurs through a complex process involving external cues (such as chemical and mechanical signals) that guide cell migration and internal mechanisms like actin cytoskeleton remodeling and myosin-driven motility. Transmembrane receptors, including integrins and CD44, allow glioma cells to adhere to and move along different components of the brain's extracellular matrix, while enzymes like matrix metalloproteinases (MMPs) degrade

surrounding tissue to clear a path for invasion. (https://www.molbiolcell.org/doi/10.1091/mbc.E18-02-0123)

Data-Set:

We will be looking mainly at the GSE62944_subsample_log2TPM and the row with data on the amounts of VEGF-A in each sample. we will need the GSE62944_metadata table to determine what cancer type each sample is.

RNA sequencing methods were used to collect data from the tumor samples. The data is taken from the Gene Expression Omnibuss and filtered by Professor Groves for class use.

The cancer types are represented with acronyms and the data

The following code shows the data extracted only from the row containing information on VEGF-A.

```python
import pandas as pd

df = pd.read_csv("GSE62944_subsample_log2TPM.csv")

vegfa_row = df[df.iloc[:, 0] == "VEGFA"].iloc[:, 1:]

print(vegfa_row)

vegfa_row.to_csv("vegfa_only.csv", index=False)
```

```
       TCGA-E9-A1NI-01A-11R-A14D-07  TCGA-E2-A1LK-01A-21R-A14D-07  \
14654                      6.169153                       6.86095

       TCGA-BH-A0B2-01A-11R-A10J-07  TCGA-E2-A107-01A-11R-A10J-07  \
14654                       5.44373                      5.012467

       TCGA-LL-A5YN-01A-11R-A28M-07  TCGA-BH-A0DQ-01A-11R-A084-07  \
14654                      4.160556                      5.434576

       TCGA-D8-A73X-01A-11R-A32P-07  TCGA-AR-A0TP-01A-11R-A084-07  \
14654                       5.38319                      9.126396

       TCGA-E2-A1IF-01A-11R-A144-07  TCGA-EW-A6SD-01A-12R-A33J-07  ...
\
14654                      5.829253                      7.556508  ...


       TCGA-N5-A4RF-01A-11R-A28V-07  TCGA-N6-A4VF-01A-31R-A28V-07  \
14654                      7.204847                       5.18758

       TCGA-N5-A4RN-01A-12R-A28V-07  TCGA-QM-A5NM-01A-11R-A28V-07  \
14654                      5.145275                       6.41909

       TCGA-N5-A4RJ-01A-11R-A28V-07  TCGA-N5-A4RO-01A-11R-A28V-07  \
14654                          5.617068                          7.726898

       TCGA-N5-A4RV-01A-21R-A28V-07  TCGA-N6-A4VD-01A-11R-A28V-07  \
14654                          6.890211                          6.707867

       TCGA-N5-A4RT-01A-11R-A28V-07  TCGA-ND-A4WC-01A-21R-A28V-07
14654                          4.810951                          8.022486

[1 rows x 1802 columns]
```

The file is very large so it is difficult to print. code must be run to turn into a csv file.

Data Analyis:

Methods

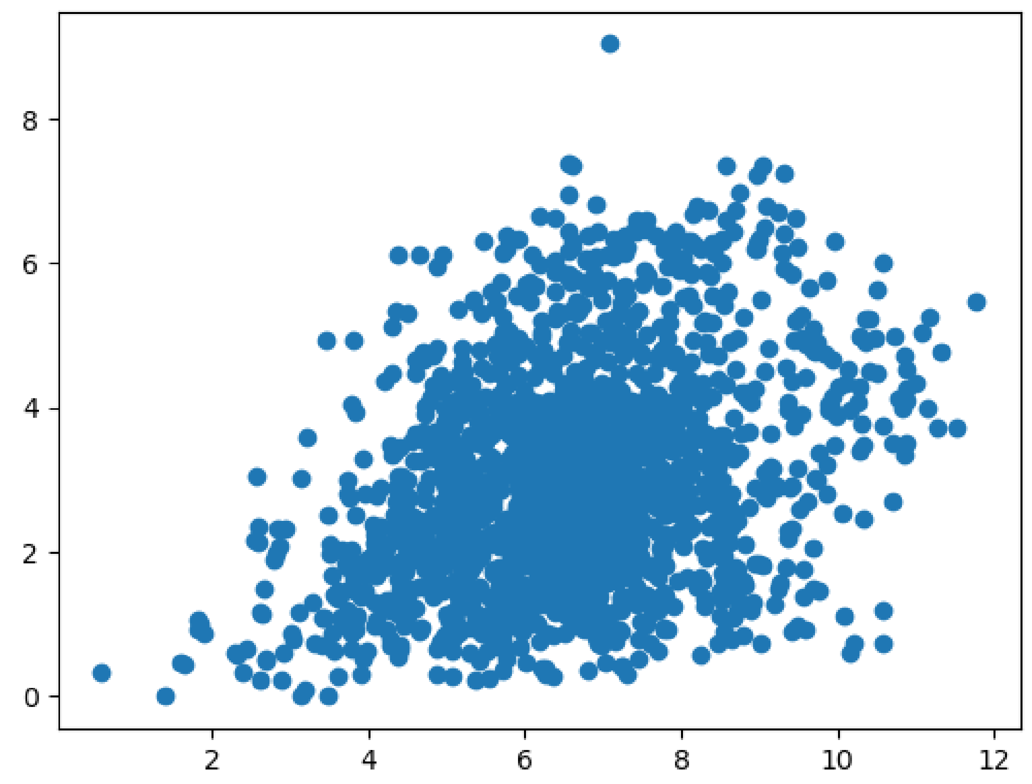
- The machine learning technique I am using is: K-means clustering. This is an unsupervised (unlabeled) ML model that is helpful when trying to find and predict patterns within the data. It requires two numerical variables, so we decided to compare VEGF-A and VEGF-C levels, two genes that directly relate to angeogenesis.
- When working with large datasets, K-Means helps break the data into smaller, easier-to-understand groups based on patterns or similarities, making it faster and more efficient to analyze.(https://www.geeksforgeeks.org/machine-learning/k-means-clustering-introduction/). This method is optimizing the distances between data points and their assigned cluster centers (centroids). This is done by taking the within-cluster sum of squares (WCSS) when adding data points to a cluster and trying to keep this WCSS number as small as possible. It takes data and forms it into neat and hopefully distinct categories. To determine whether it is "good enough," the model will try to find the best fit for the data and move around the centroids. The algorithm will eventually "give up" when reassigning points does not decrease the WCSS number. When this happens, the clusters become stable because the points are most optimized in these clusters.
- In our case, we visualized two versions of clustering (one with 4 groups and one with 20 groups), each represented by different colors to show how different cancer types distribute based on VEGF-A and VEGF-C expression levels. By comparing these two versions, we can see whether certain cancer types share similar gene expression patterns or if the patterns are more mixed across different types.

We extracted the data from the rows containing VEGF-A and VEGF-C data for each of the patients. The following code represents the raw graphs before any clustering is applied to the data

```python
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv("GSE62944_subsample_log2TPM.csv")
```

```
vegfa_row = df[df.iloc[:, 0] == "VEGFA"].iloc[:, 1:]
vegfc_row = df[df.iloc[:, 0] == "VEGFC"].iloc[:, 1:]

print("vegfa")
print(vegfa_row)
print("vegfc")
print(vegfc_row)

plt.scatter(vegfa_row, vegfc_row)
plt.plot()
plt.show()
```

```
vegfa
       TCGA-E9-A1NI-01A-11R-A14D-07 TCGA-E2-A1LK-01A-21R-A14D-07  \
14654                      6.169153             6.860950267283058

      TCGA-BH-A0B2-01A-11R-A10J-07  TCGA-E2-A107-01A-11R-A10J-07  \
14654           5.4437303908880565                      5.012467

       TCGA-LL-A5YN-01A-11R-A28M-07  TCGA-BH-A0DQ-01A-11R-A084-07  \
14654                      4.160556                      5.434576

       TCGA-D8-A73X-01A-11R-A32P-07  TCGA-AR-A0TP-01A-11R-A084-07  \
14654                       5.38319                      9.126396

       TCGA-E2-A1IF-01A-11R-A144-07  TCGA-EW-A6SD-01A-12R-A33J-07  ...
\
14654                      5.829253                      7.556508  ...


       TCGA-N5-A4RF-01A-11R-A28V-07  TCGA-N6-A4VF-01A-31R-A28V-07  \
14654                      7.204847                       5.18758

       TCGA-N5-A4RN-01A-12R-A28V-07  TCGA-QM-A5NM-01A-11R-A28V-07  \
14654                      5.145275                       6.41909

       TCGA-N5-A4RJ-01A-11R-A28V-07  TCGA-N5-A4RO-01A-11R-A28V-07  \
14654                      5.617068                      7.726898

       TCGA-N5-A4RV-01A-21R-A28V-07  TCGA-N6-A4VD-01A-11R-A28V-07  \
14654                      6.890211                      6.707867

       TCGA-N5-A4RT-01A-11R-A28V-07  TCGA-ND-A4WC-01A-21R-A28V-07
14654                      4.810951                      8.022486

[1 rows x 1802 columns]
vegfc
       TCGA-E9-A1NI-01A-11R-A14D-07 TCGA-E2-A1LK-01A-21R-A14D-07  \
14656                      3.318469             2.181462007592911

      TCGA-BH-A0B2-01A-11R-A10J-07  TCGA-E2-A107-01A-11R-A10J-07  \
14656           3.698131606356658                           3.563565

       TCGA-LL-A5YN-01A-11R-A28M-07  TCGA-BH-A0DQ-01A-11R-A084-07  \
14656                      2.900182                      4.566632

       TCGA-D8-A73X-01A-11R-A32P-07  TCGA-AR-A0TP-01A-11R-A084-07  \
14656                      3.615998                      3.040907

       TCGA-E2-A1IF-01A-11R-A144-07  TCGA-EW-A6SD-01A-12R-A33J-07  ...
\
14656                      4.635237                      3.607043  ...


       TCGA-N5-A4RF-01A-11R-A28V-07  TCGA-N6-A4VF-01A-31R-A28V-07  \
14656                      2.695069                      4.810356

       TCGA-N5-A4RN-01A-12R-A28V-07  TCGA-QM-A5NM-01A-11R-A28V-07  \
14656                       4.35217                       2.60216

       TCGA-N5-A4RJ-01A-11R-A28V-07  TCGA-N5-A4RO-01A-11R-A28V-07  \
14656                      2.905986                      2.684326

       TCGA-N5-A4RV-01A-21R-A28V-07  TCGA-N6-A4VD-01A-11R-A28V-07  \
14656                      2.391044                      3.779163

       TCGA-N5-A4RT-01A-11R-A28V-07  TCGA-ND-A4WC-01A-21R-A28V-07
14656                      3.577317                      2.649205

[1 rows x 1802 columns]

/var/folders/0n/kr0rzp5n1xs8b1cx6yv17vc80000gn/T/
ipykernel_1914/1374783230.py:8: DtypeWarning: Columns (2,3) have mixed
types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("GSE62944_subsample_log2TPM.csv")
```

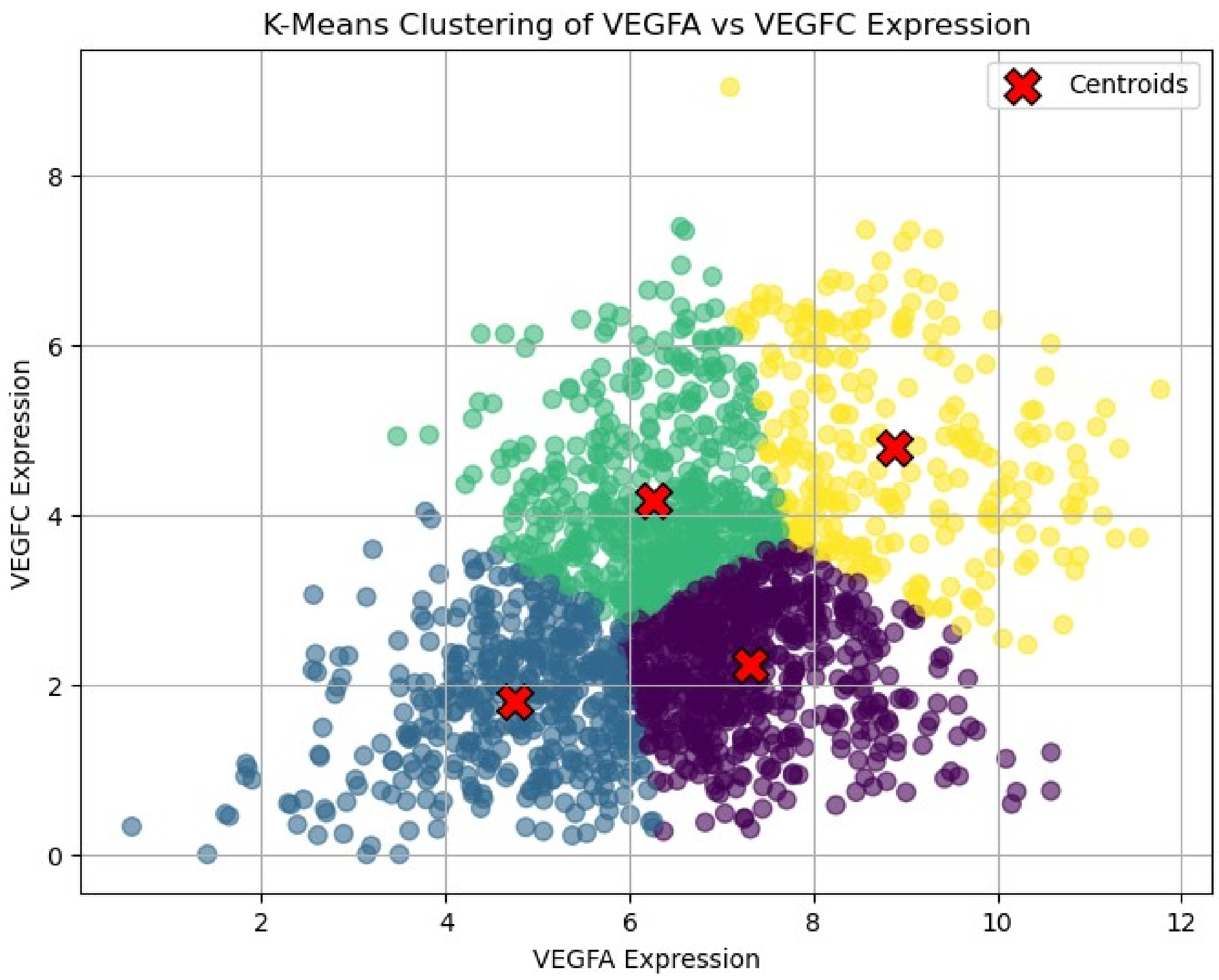


The following code displays our data using clustering with 4 groups. We initially wanted to test the code utilizing a smaller number of groups to determine how the custering would create the different clusters when a kmeans clustering method is used. The graph displays the different clusters in different colors and uses a red x to show the centroid of each cluster.

```
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("GSE62944_subsample_log2TPM.csv")

vegfa_row = df[df.iloc[:, 0] == "VEGFA"].iloc[:, 1:]
vegfc_row = df[df.iloc[:, 0] == "VEGFC"].iloc[:, 1:]

vegfa_values = vegfa_row.to_numpy().flatten()
vegfc_values = vegfc_row.to_numpy().flatten()


X = np.column_stack((vegfa_values, vegfc_values))

kmeans = KMeans(n_clusters=4, random_state=0, n_init="auto")
kmeans.fit(X)

labels = kmeans.labels_
```

```
centers = kmeans.cluster_centers_


# graph clustering
plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50,
alpha=0.6)

plt.scatter(
    centers[:, 0], centers[:, 1],
    c='red', s=200, marker='X', edgecolor='k', label='Centroids'
)

plt.xlabel("VEGFA Expression")
plt.ylabel("VEGFC Expression")
plt.title("K-Means Clustering of VEGFA vs VEGFC Expression")
plt.legend()
plt.grid(True)
plt.show()
```

```
/var/folders/0n/kr0rzp5n1xs8b1cx6yv17vc80000gn/T/
ipykernel_1914/3499897745.py:7: DtypeWarning: Columns (2,3) have mixed
types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("GSE62944_subsample_log2TPM.csv")
```

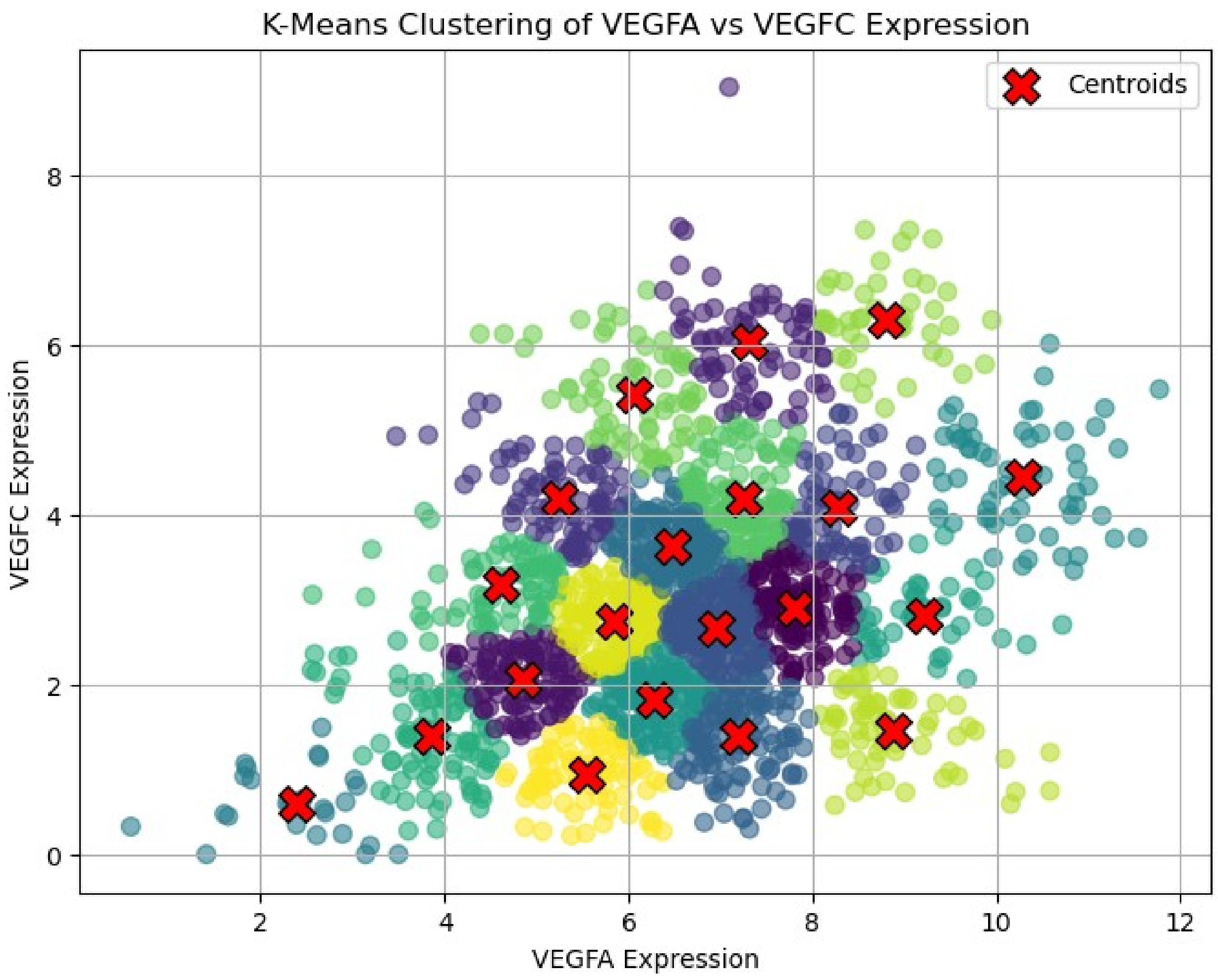


Next, we changed the code to have 20 clusters (closer number to the number of different cancer types included in the dataset)

```python
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("GSE62944_subsample_log2TPM.csv")

vegfa_row = df[df.iloc[:, 0] == "VEGFA"].iloc[:, 1:]
vegfc_row = df[df.iloc[:, 0] == "VEGFC"].iloc[:, 1:]

vegfa_values = vegfa_row.to_numpy().flatten()
vegfc_values = vegfc_row.to_numpy().flatten()


X = np.column_stack((vegfa_values, vegfc_values))
```

```python
kmeans = KMeans(n_clusters=20, random_state=0, n_init="auto")
kmeans.fit(X)

labels = kmeans.labels_
centers = kmeans.cluster_centers_


# graph clustering
plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50,
alpha=0.6)

plt.scatter(
    centers[:, 0], centers[:, 1],
    c='red', s=200, marker='X', edgecolor='k', label='Centroids'
)

plt.xlabel("VEGFA Expression")
plt.ylabel("VEGFC Expression")
plt.title("K-Means Clustering of VEGFA vs VEGFC Expression")
plt.legend()
plt.grid(True)
plt.show()
```

```
/var/folders/0n/kr0rzp5n1xs8b1cx6yv17vc80000gn/T/
ipykernel_1914/3258036389.py:7: DtypeWarning: Columns (2,3) have mixed
types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("GSE62944_subsample_log2TPM.csv")
```



Verify and validate your analysis:

Silhouette coefficient is comparing the distance between clusters with the distance within clusters. The closer the point within a cluster are to each other and the farther each cluster is from each other, the better the clustering. if score = 0: mid scoring if score < 0: bad scoring (lots of variance and overlapping) if score > 0: good clustering.

The following code shows the silhouette score for the sample when using 4, 8, 16, 20, and 23 clusters as well as the graph generated for each clustering numbers.

```
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm
```

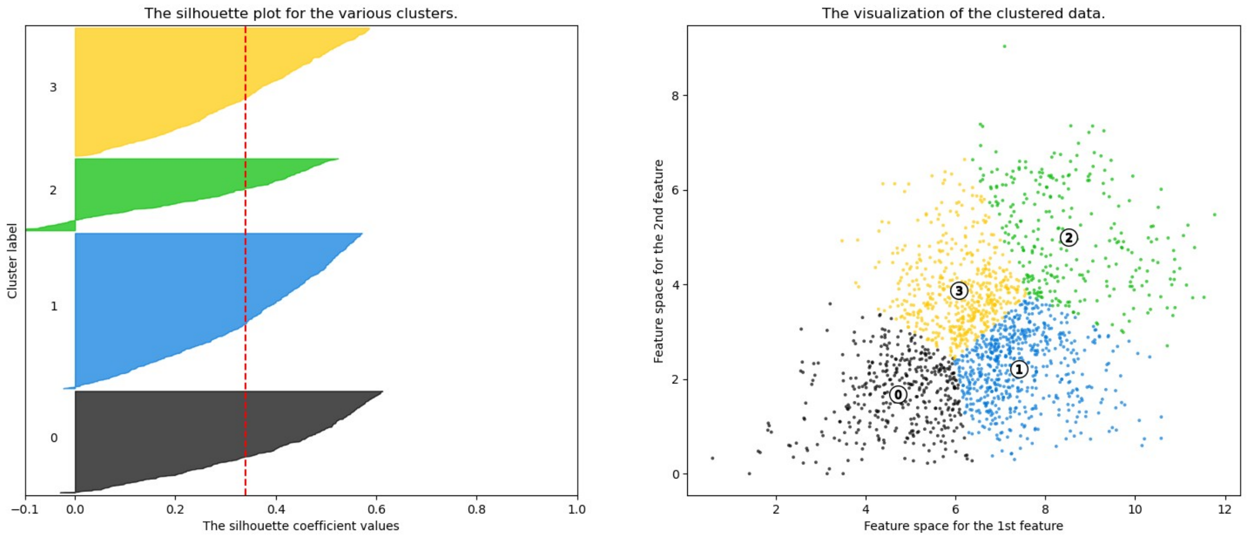
```python
df = pd.read_csv("GSE62944_subsample_log2TPM.csv")

vegfa_row = df[df.iloc[:, 0] == "VEGFA"].iloc[:, 1:]
vegfc_row = df[df.iloc[:, 0] == "VEGFC"].iloc[:, 1:]

vegfa_values = vegfa_row.to_numpy().flatten()
vegfc_values = vegfc_row.to_numpy().flatten()


X = np.column_stack((vegfa_values, vegfc_values))

kmeans = KMeans(n_clusters=4, random_state=0, n_init="auto")
kmeans.fit(X)



range_n_clusters = [4, 8, 16, 20, 23]

for n_clusters in range_n_clusters:
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)


    ax1.set_xlim([-0.1, 1])

    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])


    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(X)


    silhouette_avg = silhouette_score(X, cluster_labels)
    print(
        "For n_clusters =",
        n_clusters,
        "The average silhouette_score is :",
        silhouette_avg,
    )

    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):

        ith_cluster_silhouette_values =
sample_silhouette_values[cluster_labels == 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)
        ax1.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
        ax1.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  # 10 for the 0 samples

    ax1.set_title("The silhouette plot for the various clusters.")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # The vertical line for average silhouette score of all the values
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd Plot showing the actual clusters formed
    colors = cm.nipy_spectral(cluster_labels.astype(float) /
n_clusters)
    ax2.scatter(
        X[:, 0], X[:, 1], marker=".", s=30, lw=0, alpha=0.7, c=colors,
edgecolor="k"
    )

    # Labeling the clusters
    centers = clusterer.cluster_centers_
    # Draw white circles at cluster centers
    ax2.scatter(
        centers[:, 0],
        centers[:, 1],
        marker="o",
        c="white",
        alpha=1,
        s=200,
        edgecolor="k",
    )

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker="$%d$" % i, alpha=1, s=50,
edgecolor="k")

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(
        "Silhouette analysis for KMeans clustering on sample data with
n_clusters = %d"
        % n_clusters,
        fontsize=14,
        fontweight="bold",
    )

plt.show()
```

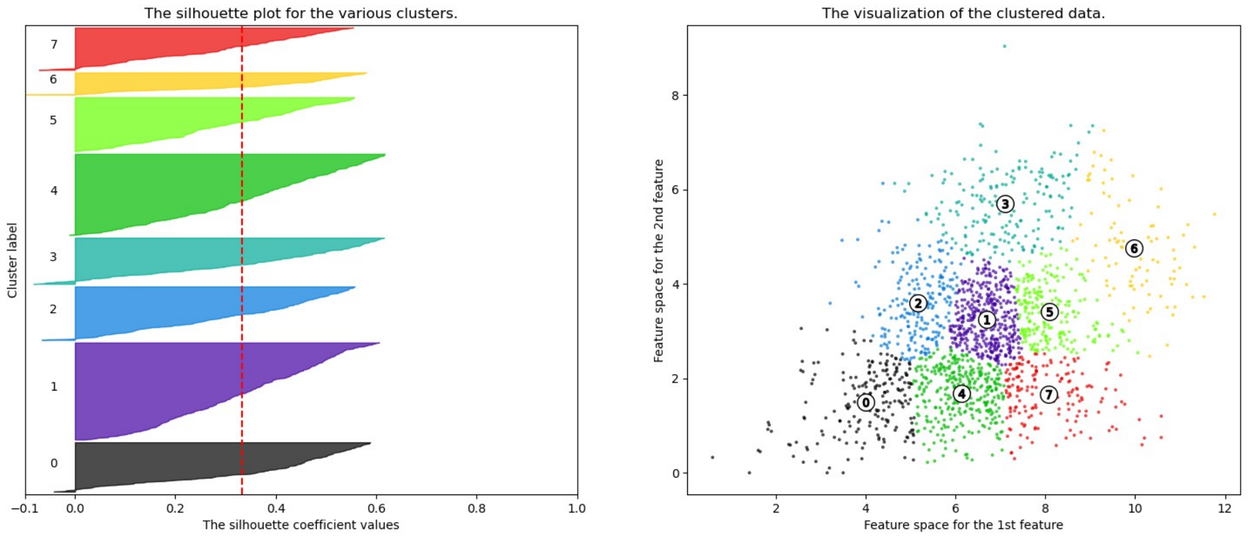
```
/var/folders/0n/kr0rzp5n1xs8b1cx6yv17vc80000gn/T/
ipykernel_1644/2735650792.py:13: DtypeWarning: Columns (2,3) have
mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("GSE62944_subsample_log2TPM.csv")
```

```
For n_clusters = 4 The average silhouette_score is :
0.3398960817549879
For n_clusters = 8 The average silhouette_score is :
0.33344313938549097
For n_clusters = 16 The average silhouette_score is :
0.34214916853237687
For n_clusters = 20 The average silhouette_score is :
0.33703037985531614
For n_clusters = 23 The average silhouette_score is :
0.33154642679615953
```

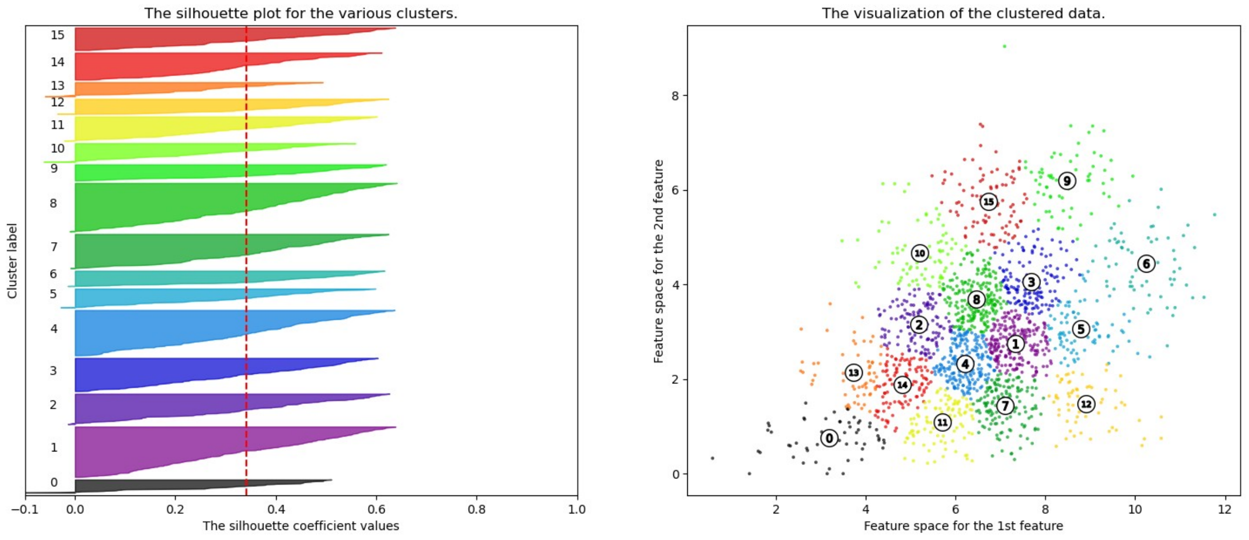
Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



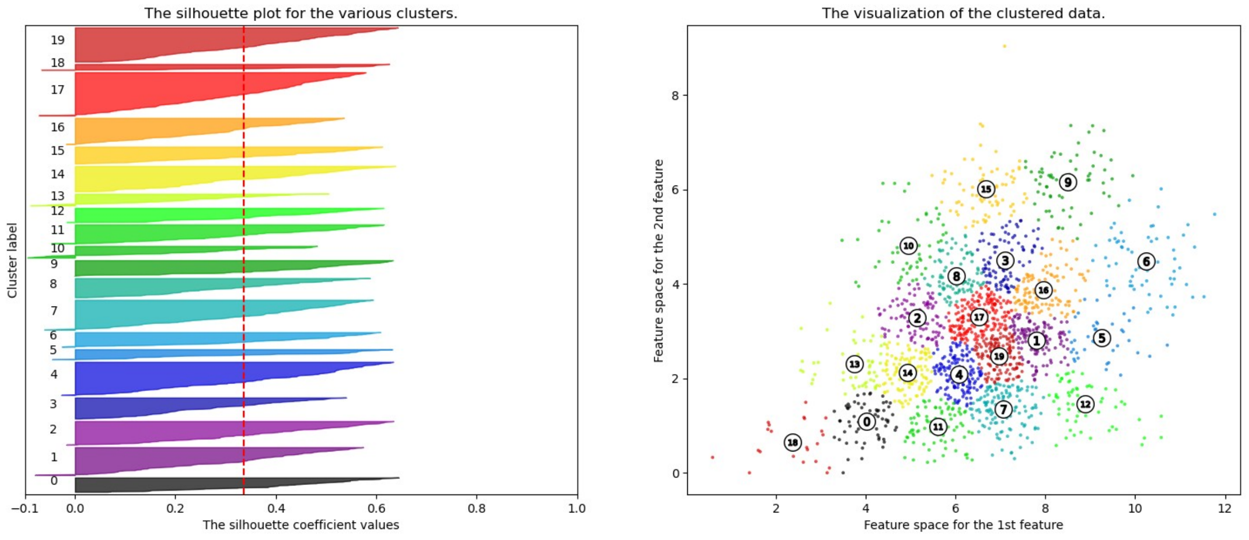
Silhouette analysis for KMeans clustering on sample data with n_clusters = 8

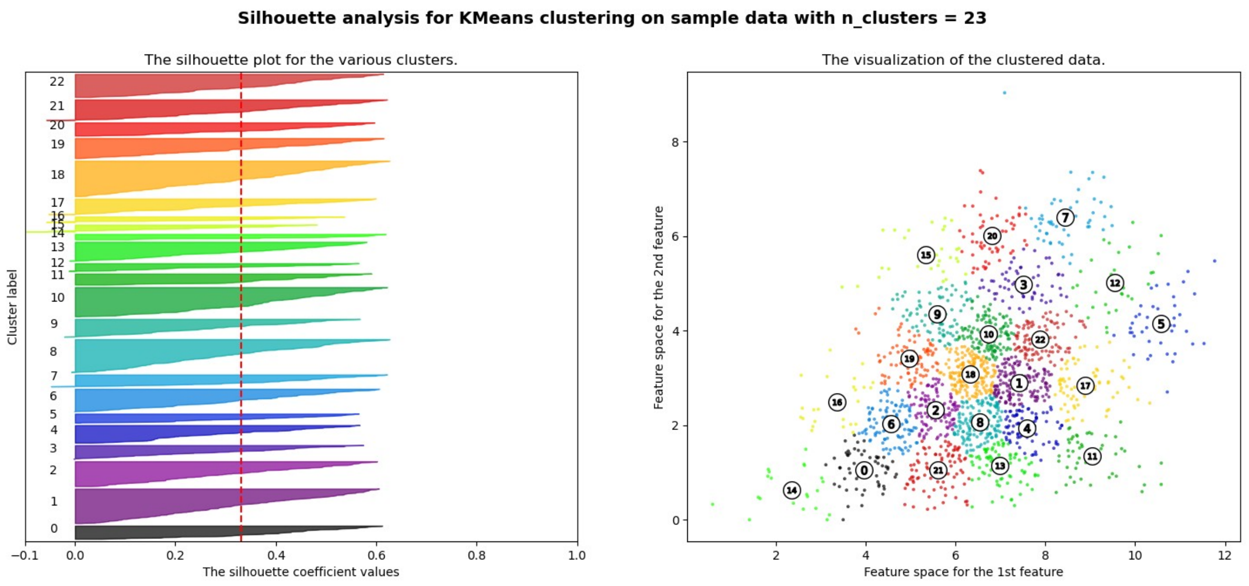


Silhouette analysis for KMeans clustering on sample data with n_clusters = 16



Silhouette analysis for KMeans clustering on sample data with n_clusters = 20





The silhouette score for when 20 clusters are used is 0.322. This score is not significantly different from that of the score with 4 clusters. This is suggesting that VEGFA and VEGFC expression does not have significant differences in the different cancer types as if there were, there would be more distinct clusters with higher silhouette scores as we come closer to the true number of different cancers represented in the dataset. There are closer to 20 cancers than 4 cancers represented in the dataset and as such, we would expect the silhouette score to increase between 4 to 20 cancers. The opposite is shown with the silhouette score slightly decreasing as the number of cancers increases, suggesting to us that VEGFA and VEGFC levels do not impact the type of cancer exhibited in an individual.

The following code splits the data in half where the first half of the data is used to train the data and the second half is used to test the dataset with 4 clusters.

```python
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm
from sklearn.model_selection import train_test_split


df = pd.read_csv("GSE62944_subsample_log2TPM.csv")

vegfa_row = df[df.iloc[:, 0] == "VEGFA"].iloc[:, 1:]
vegfc_row = df[df.iloc[:, 0] == "VEGFC"].iloc[:, 1:]

vegfa_values = vegfa_row.to_numpy().flatten()
vegfc_values = vegfc_row.to_numpy().flatten()
```

```
X = np.column_stack((vegfa_values, vegfc_values))

X_train, X_test = train_test_split(X, test_size=0.5, random_state=0)


kmeans = KMeans(n_clusters=4, random_state=0, n_init="auto")
kmeans.fit(X_train)

train_labels = kmeans.predict(X_train)
test_labels = kmeans.predict(X_train)

if len(np.unique(train_labels)) > 1:
    train_silhouette = silhouette_score(X_train, train_labels)
    print(f"Train Silhouette Score: {train_silhouette:.2f}")

if len(np.unique(test_labels)) > 1:
    test_silhouette = silhouette_score(X_test, test_labels)
    print(f"Test Silhouette Score: {test_silhouette:.2f}")
```

```
Train Silhouette Score: 0.34
Test Silhouette Score: -0.03
```

```
/var/folders/0n/kr0rzp5n1xs8b1cx6yv17vc80000gn/T/
ipykernel_1644/2048083487.py:13: DtypeWarning: Columns (2,3) have
mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("GSE62944_subsample_log2TPM.csv")
```

The following code splits the data in half where the first half of the data is used to train the data and the second half is used to test the dataset with 20 clusters.

```
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm
from sklearn.model_selection import train_test_split


df = pd.read_csv("GSE62944_subsample_log2TPM.csv")

vegfa_row = df[df.iloc[:, 0] == "VEGFA"].iloc[:, 1:]
vegfc_row = df[df.iloc[:, 0] == "VEGFC"].iloc[:, 1:]

vegfa_values = vegfa_row.to_numpy().flatten()
```

```
vegfc_values = vegfc_row.to_numpy().flatten()


X = np.column_stack((vegfa_values, vegfc_values))

X_train, X_test = train_test_split(X, test_size=0.5, random_state=0)


kmeans = KMeans(n_clusters=20, random_state=0, n_init="auto")
kmeans.fit(X_train)

train_labels = kmeans.predict(X_train)
test_labels = kmeans.predict(X_train)

if len(np.unique(train_labels)) > 1:
    train_silhouette = silhouette_score(X_train, train_labels)
    print(f"Train Silhouette Score: {train_silhouette:.2f}")

if len(np.unique(test_labels)) > 1:
    test_silhouette = silhouette_score(X_test, test_labels)
    print(f"Test Silhouette Score: {test_silhouette:.2f}")
```

```
Train Silhouette Score: 0.34
Test Silhouette Score: -0.13

/var/folders/0n/kr0rzp5n1xs8b1cx6yv17vc80000gn/T/
ipykernel_1644/358834778.py:13: DtypeWarning: Columns (2,3) have mixed
types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("GSE62944_subsample_log2TPM.csv")
```

The training silhouette score is decent whereas the test silhouette score is much worse. this is indicating that our model is overfitting the data.

External validation/invalidation of silhouette results:

- "We found that larger (or smaller) AS values agreed well with both higher (or lower) degrees of separation between different groups and higher percentages of differentially expressed genes (PDEG)." (https://biologicalproceduresonline.biomedcentral.com/articles/10.1186/s12575-018-0067-8?utm_source=chatgpt.com)
  - This source claims that the higher the silhouette score(better clusters), the genes are more differentially expressed. Our relatively low train and test silhouette scores indicate that the expressions for VEGF-A and VEGF-C are not distinct.
  - Our results can be partially validated by source (https://pubmed.ncbi.nlm.nih.gov/10571529/). They found that VEGF-A and VEGF-C show partially overlapping binding patterns in embryonic tissues, but in adult tissues they have distinct binding sites.
- Another article we found looks at the VEGF family across various cancer types using the Wilcoxon signed-rank test. "For a few cancers, overexpression of VEGF family genes and

their correlation with the prognosis, metastasis, and recurrence have been reported. Compared to low expression, high VEGFA expression is associated with poor survival outcomes in gastric cancer [3], lung cancer [4], and colon cancer [5]. VEGFB facilitates tumor advancement by elevating plasminogen activators, which can lead to the metastasis of breast cancer [6]. The expression of VEGFC and VEGFD correlates with recurrence in head and neck squamous cell carcinomas [7] and lymphatic metastases in gastric cancer [8], respectively." (https://pubmed.ncbi.nlm.nih.gov/37852616/)
    - These findings directly contradict our silhouette test results. When we compared the silhouette score of the cluster size of 4 and that of cluster size 20, we did not find a singnificant difference (indicating that these gene expressions do not significantly differ with different cancer types). However, the article presents solid evidence that these VEGF genes are differentially expressed across cancer types and stages, invalidating our results.
- We took another look at the VEGF family, but this time at the biological differences in the roles that each gene plays in the article: https://pmc.ncbi.nlm.nih.gov/articles/PMC551528/?utm_source=chatgpt.com. Certain genes like VEGF-C and VEGF-D are tied to lymphangiogenesis, while others like VEGF-A are more important for angiogenesis. The article also emphasizes that VEGF family members have distinct receptor-binding profiles and specialized roles rather than being fully redundant. All in all it supports that VEGF-A and VEGF-C are distinct. They have different primary functions (blood vessel vs lymphatic/dual), different receptor affinities, different isoform/processing dynamics. That means differences in their expression profiles can reflect different biology, which is relevant when clustering samples by their expression.
    - This also invalidates our results because we did not get distinct clusters for VEGF-A and VEGF-C expression. We claimed that there is quite a bit of overlap between the genes based on our silhouette scores. This is contradicts the research that suggests that these genes have distinct functions in the body and as it relates to cancer.

Conclusions and Ethical Implications:

(Think about the answer your analysis generated, draw conclusions related to your overarching question, and discuss the ethical implications of your conclusions.

Limitations and Future Work:

(Think about the answer your analysis generated, draw conclusions related to your overarching question, and discuss the ethical implications of your conclusions.

NOTES FROM YOUR TEAM:

- hallmark: angiogensis
- we are focusing on patterns across multiple cancer types
- we will use metadata to make predictions
- we will mainly look at the VEGF-A protein and VEGF-C proteins and use clustering to group them

- our goal is to use clustering to see if there are groups of VEGFA and VEGFC proteins (both contributing to angiogenesis in patients) for each cancer group. If there are distinct VEGFA and VEGFC levels for each cancer group, it is indicating that VEGFA and VEGFC levels differ between cancer groups and those protein levels can be used to predict the cancer type an individual will develop.

QUESTIONS FOR YOUR TA:

We have no questions for our TA