# Module_3: *(Template)*

## Team Members:

Daniel Garcia-Soliz, Trey Hancock

## Project Title:

Hallmarks of Cancer

## Project Goal:

This project seeks to analyze the correlation between major angiogenesis related biomarkers and compare which cancers have the most prevelance with this issue.

## 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: Focus will be on angiogenisis. Specifically, investigating the pathway involved with exciting the process in support of specific cancer cells and comparing the cancers biomarkers.
- Overview of hallmark:
  - Angiogenesis is defined by the growth of blood vessels throughout the body to support a high demand of waste management, oxygen and nutrient defficiency, etc. Cancer and proliferating cells seem to increase the rate of this growth during the mid-life phase of tumor propogation. This is typically prior to excessive tumor growth and is named the "angiogenic switch."
  - Sources: Hallmarks of Cancer by Hanahan and Weinburg
- Genes associated with hallmark to be studied (describe the role of each gene, signaling pathway, or gene set you are going to investigate):
  - Cancer cells typically are associated with angiogenic activators such as VEGF, bFGF, angiogenin, TGF-alpha and beta, etc. Specifically the VEGF family they are commonly seen in cancerous tissues as a powerful angiogenic agent. It takes advantage of this by feeding into the into the new blood vessels and secreting itself into the newly revealed tissue. Due to all this, angiogenesis factors play a major role in the spread of tumors.

    - Sources: https://pmc.ncbi.nlm.nih.gov/articles/PMC1993983/ |

- Prevalence & incidence
    - Angiogenesis seems to be most involved in fast spreading tumors. Examples of these cancers are breast, melanoma, colorectal, and cell lung cancer. All these can classify under malignant tumors from their invassive hallmarks. To help give more prevalence and incidence statistics, we will determine malignant tumors to be effected by the angiogenesis hallmark. Just this year there are nearly 2,000,000 cases of malignant tumors, with the numbers spreading to 320,000 breast cancer, 150,000 colonrectal, 100,000 melanoma cancer cases and more.
    - Sources:https://www.cancer.gov/about-cancer/understanding/statistics?utm_source=chatgpt.com and https://www.nature.com/articles/s41392-023-01460-1
- Risk factors (genetic, lifestyle) & Societal determinants
    - Certain cancers have specific lifestyle and genetic risk factors that differentiate them, regardless of their malignant type. For lung cancer many factors associated with smoking, chemical exposure, and family history can increase the risk. Overlapping risks with breast cancer is seen to be drug/alcohol excessive use or physical inactivity. However, these risk factors seem to also overlap with other common malignant tumors.
    - Sources: https://www.missouribaptist.org/Medical-Services/Cancer-Care/Cancer-Post/ArtMID/527/ArticleID/2852/Did-You-Know-The-Overlap-Between-Lung-and-Breast-Cancer
- Standard of care treatments (& reimbursement)
    - Regardless of the cancer, it is a possibility to target angiogenesis within quick spreading cancer. It is possible to target the VEGF families with drugs for blocks. Furthmore, "Tyrosine Kinase Inhibitors" are an advancing medicine for blocking multiple angiogenesis pathways.
    - Citation: OpenAI. (2025, October 27). Anti-angiogenic therapy in cancer (ChatGPT response). ChatGPT (GPT-5) [Large language model]. https://chat.openai.com/
- Biological mechanisms (anatomy, organ physiology, cell & molecular physiology)
    - As mentioned prior, the angiogenic switch begins the tumor spread to further tissues. This is when pro-angiogenic factors are expressed for VEGF family interaction.
    - The tumor vessels being more chaotic and leaky provides an easiser invasion for spreading angiogenic factors in nearby tissue. Sources: https://pubmed.ncbi.nlm.nih.gov/21376230/

## Data-Set:

The data analyzed in this study were obtained from The Cancer Genome ATlas (TCGA) and reprocessed as part of the publicly available GSE62944 dataset, described by Rahman et al. (2015) in "Alternative preprocessing of RNA-Sequencing data in The Cancer Genome Atlas leads to improved analysis results". The dataset encompasses 9,264 tumor samples and 741 matched normal tissues, representing 24 distinct cancer types.

All sequencing was performed on the Illumina HiSeq 2000 platform, generating 101-base-pair reads from tumor and normal tissue samples. The raw reads were aligned to the UCSC hg19 human reference genome using the Rsubread algorithm, which improves accuracy when mapping across exon junctions. Gene counts were normalized using the Transcripts Per Million method. These values were then $\log_2$-transformed so that expression could be directly compared across samples and cancer types.

Each sample also includes clinical information such as cancer type, tumor stage, and patient demographics.

Our project will focus on three genes: VEGFA, VEGFB, and VEGFC. These genes are major regulators of angiogenesis. With regard to the clinical information, we will specifically look at different cancer types to identify which tumors exhibit the strongest angiogenic activity.

## Data Analyis:

### Methods

The machine learning technique we are using is: Unsupervised Clustering.

Specifically, we will use k-means clustering applied to the expression levels of angiogenesis-related genes VEGFA, VEGFB, and VEGFC across all cancer types in the GSE62944 dataset. The goal of clustering is to group the samples into clusters where the samples inside the same cluster are more similar to each other than to samples in other clusters. We are using this approach because we want to see whether cancers can be grouped based on their VEGF expression levels, even if they come from completely different cancer types.

This method is a good fit because our research question asks whether cancers show distinct patterns of VEGF expression and whether any cancers share similar angiogenesis biomarkers. Clustering enables us to identify groups of tumor samples that have similar values for VEGFA, VEGFB, and VEGFC expression. Unlike supervised learning, which requires known labels, unsupervised clustering helps us discover biological patterns that may not align with clinical categories.

K-means clustering optimizes the within-cluster sum of squared distances. This basically means how well the samples fit within their assigned clusters. The algorithm stars by placing centers, and then it assigns each tumor sample to whichever centroid it is closest to. Then, the algorithm moves each centroid to the average position of samples assigned to it.

Because clustering is unsupervised, there is no accuracy score or correct answer to compare against. Instead, we evaluate model quality based on how good the clustering structure is internally. We could use the silhouette score, which would tell us how similar a sample is to its own cluster compared to other clusters.

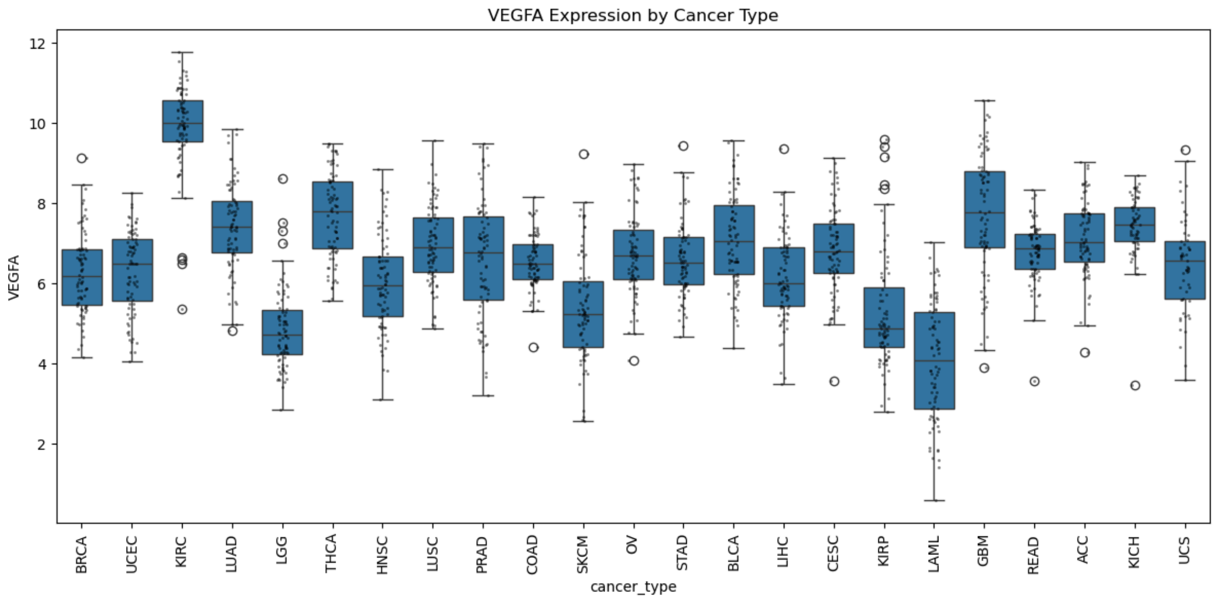
## Analysis

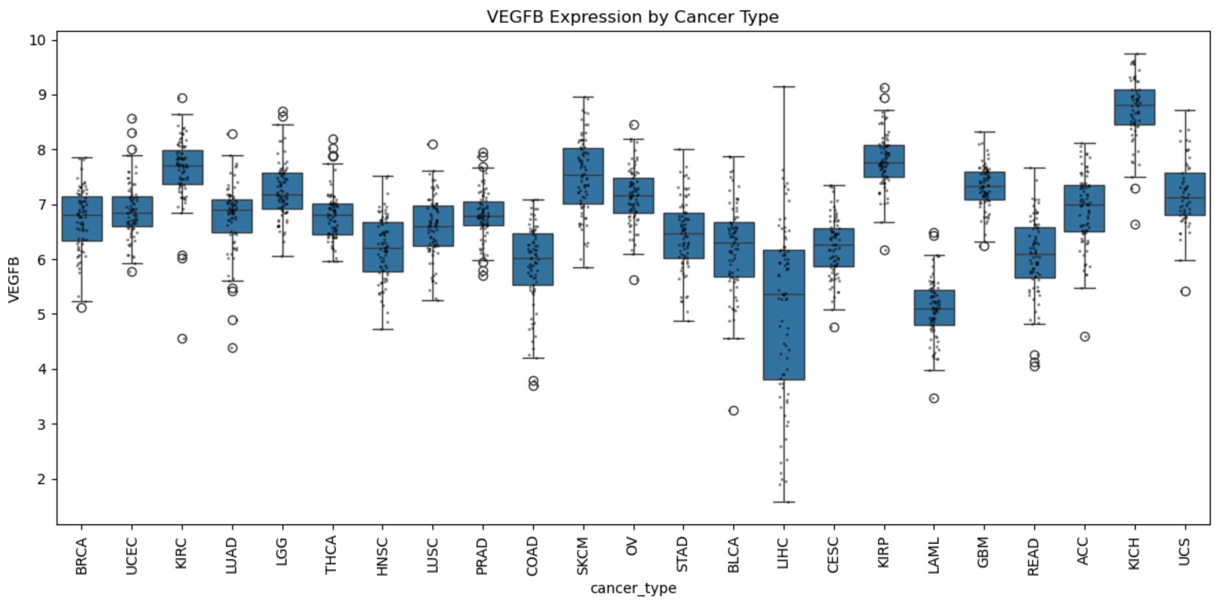
*(Describe how you analyzed the data. This is where you should intersperse your Python code so that anyone reading this can run your code to perform the analysis that you did, generate your figures, etc.)*

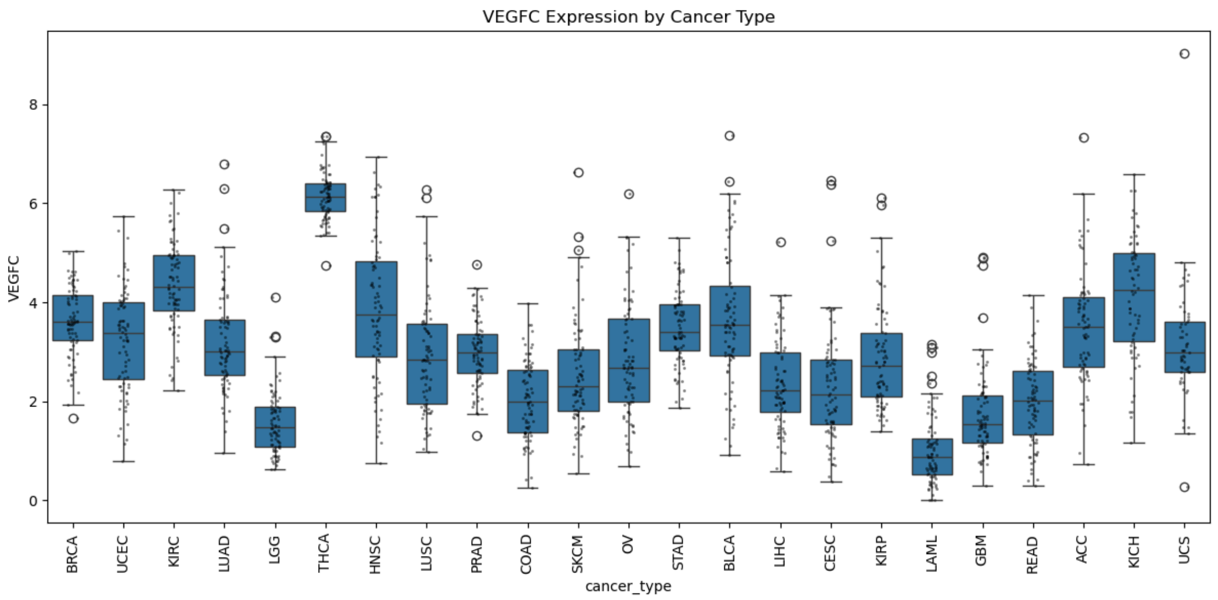
In [ ]:
```
# =========================
# IMPORTS & CONSTANTS
# =========================
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from itertools import combinations
from scipy import stats
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    silhouette_score, silhouette_samples, adjusted_rand_score, confusion_matrix
)

# File paths
EXPR_FILE = "GSE62944_subsample_log2TPM.csv"   # expression matrix: genes x samples
META_FILE = "GSE62944_metadata.csv"            # metadata with at least ['sample','
GENES = ["VEGFA", "VEGFB", "VEGFC"]            # angiogenesis panel

RESULTS_DIR = "vegf_results"
os.makedirs(RESULTS_DIR, exist_ok=True)

# =========================
# DATA LOADING
# =========================
def load_data(expr_path, meta_path):
    expr = pd.read_csv(expr_path, index_col=0)
    meta = pd.read_csv(meta_path)
    # Align by common samples to avoid misalignment
    common_samples = [s for s in expr.columns if s in meta["sample"].values]
    expr = expr[common_samples]
    meta = meta[meta["sample"].isin(common_samples)]
    return expr, meta

# =========================
# PER-GENE DESCRIPTIVES & ANOVA
# =========================
def compute_statistics(expr, meta):
    df = expr.loc[GENES].T
    df = df.merge(meta[["sample", "cancer_type"]], left_index=True, right_on="sampl
    for gene in GENES:
        summary = df.groupby("cancer_type")[gene].agg(["mean", "std", "count"])
        summary = summary.sort_values(by="mean", ascending=False)
        print(f"\n{gene} Expression Summary (mean ± std, count), sorted by mean:\n"
        print(summary)
        groups = [grp[gene].values for _, grp in df.groupby("cancer_type")]
        f_val, p_val = stats.f_oneway(*groups)
        print(f"\n{gene} ANOVA F={f_val:.2f}, p={p_val:.3e}")

# =========================
# BOXPLOTS BY CANCER TYPE
# =========================
def plot_expression_by_cancer_type(expr, meta):
    df = expr.loc[GENES].T
    df = df.merge(meta[["sample", "cancer_type"]], left_index=True, right_on="sampl
    for gene in GENES:
        plt.figure(figsize=(12, 6))
        sns.boxplot(x="cancer_type", y=gene, data=df)
        sns.stripplot(x="cancer_type", y=gene, data=df, color="black", size=2, alph
        plt.title(f"{gene} Expression by Cancer Type")
        plt.xticks(rotation=90)
        plt.tight_layout()
        plt.show()

# =========================
# PCA OVERVIEW (ALL CANCERS, VEGF GENES ONLY)
# =========================
def plot_pca(expr, meta):
    df = expr.loc[GENES].T
    cancer_types = meta.set_index("sample").loc[df.index, "cancer_type"]
    pca = PCA(n_components=2)
    pca_result = pca.fit_transform(df)
    plt.figure(figsize=(8, 6))
    sns.scatterplot(
        x=pca_result[:, 0],
        y=pca_result[:, 1],
        hue=cancer_types,
        palette="tab20",
        s=60,
        alpha=0.8,
    )
    plt.title("PCA of VEGF Gene Expression")
    plt.xlabel(f"PC1 ({pca.explained_variance_ratio_[0]*100:.1f}% variance)")
    plt.ylabel(f"PC2 ({pca.explained_variance_ratio_[1]*100:.1f}% variance)")
    plt.legend(bbox_to_anchor=(1.05, 1), loc="upper left")
    plt.tight_layout()
    plt.show()

# =========================
# EXPLORATORY: K-MEANS ON ALL CANCERS (UNVALIDATED)
# =========================
def plot_cluster_composition(expr, meta):
    df = expr.loc[GENES].T
    kmeans = KMeans(n_clusters=3, random_state=42, n_init=20)
    clusters = kmeans.fit_predict(df)
    meta = meta.copy()
    meta["cluster"] = clusters
    plt.figure(figsize=(8, 6))
    sns.countplot(x="cluster", hue="cancer_type", data=meta, palette="tab20")
    plt.title("Cluster Composition by Cancer Type (Exploratory)")
    plt.xlabel("Cluster")
    plt.ylabel("Number of Samples")
    plt.legend(bbox_to_anchor=(1.05, 1), loc="upper left")
    plt.tight_layout()
    plt.show()

# ==========================
# HELPERS FOR 3-CANCER VALIDATED PIPELINE
# ==========================
def _pca_on_vegf_all(expr, meta, random_state=42):
    """PCA on all samples (VEGFA/B/C), after standardization, for farthest-3 select
    X = expr.loc[GENES].T.values
    X_std = StandardScaler().fit_transform(X)
    pca = PCA(n_components=2, random_state=random_state)
    coords = pca.fit_transform(X_std)
    df = pd.DataFrame(coords, index=expr.columns, columns=["PC1", "PC2"])
    df["sample"] = df.index
    df = df.merge(meta[["sample", "cancer_type"]].drop_duplicates("sample"), on="sa
    return df

def _select_farthest_3_cancers(pca_df, min_per_type=15):
    """
    Choose the 3 cancer types with the largest mean pairwise centroid distance in P
    requiring at least min_per_type samples per type.
    """
    cent = pca_df.groupby("cancer_type")[["PC1","PC2"]].agg(["mean","count"])
    cent.columns = [f"{a}_{b}" for a,b in cent.columns]
    cent = cent.rename(columns={"PC1_mean":"x","PC2_mean":"y","PC1_count":"n"})
    pool = cent[cent["n"] >= min_per_type]
    names = pool.index.tolist()
    if len(names) < 3:
        raise ValueError(f"Need ≥3 cancer types with at least {min_per_type} sample
    best, best_m = None, -1.0
    for a, b, c in combinations(names, 3):
        pts = np.array([[pool.loc[a,"x"], pool.loc[a,"y"]],
                        [pool.loc[b,"x"], pool.loc[b,"y"]],
                        [pool.loc[c,"x"], pool.loc[c,"y"]]])
        d = [np.linalg.norm(pts[i]-pts[j]) for i,j in [(0,1),(0,2),(1,2)]]
        m = float(np.mean(d))
        if m > best_m:
            best, best_m = (a,b,c), m
    return list(best)

def _majority_label_map(train_clusters, y_train):
    """
    For each train cluster id, assign the majority true label (cancer_type).
    Used to convert unsupervised clusters into pseudo-class predictions for accurac
    """
    train_clusters = np.asarray(train_clusters)
    y_train = np.asarray(y_train)
    mapping = {}
    for c in np.unique(train_clusters):
        mask = (train_clusters == c)
        vals, counts = np.unique(y_train[mask], return_counts=True)
        mapping[int(c)] = vals[np.argmax(counts)]
    return mapping

def _plot_silhouette(X_std, labels, title, out_png):
    """Silhouette plot (per-sample) for a given feature matrix and cluster labels."
    sil_avg = silhouette_score(X_std, labels)
    sil_values = silhouette_samples(X_std, labels)
    n_clusters = len(np.unique(labels))
    plt.figure(figsize=(8, 6))
    y_lower = 10
    for i in range(n_clusters):
        ith = sil_values[labels == i]
        ith.sort()
        size_i = ith.shape[0]
        y_upper = y_lower + size_i
        plt.fill_betweenx(np.arange(y_lower, y_upper), 0, ith, alpha=0.7)
        plt.text(-0.05, y_lower + 0.5 * size_i, str(i))
        y_lower = y_upper + 10
    plt.axvline(x=sil_avg, linestyle="--")
    plt.title(title + f" (avg={sil_avg:.3f})")
    plt.xlabel("Silhouette coefficient")
    plt.ylabel("Cluster")
    plt.tight_layout()
    plt.savefig(out_png)
    plt.show()

def _focused_pca_scatter(expr_sub, meta_sub, clusters, genes, title, out_png):
    """
    PCA on the 3-cancer subset. Color by cancer_type, style by cluster.
    """
    df = expr_sub.T.copy()
    df["cluster"] = clusters
    df["sample"] = df.index
    df = df.merge(meta_sub[["sample","cancer_type"]], on="sample", how="left")
    pca = PCA(n_components=2, random_state=42)
    pcs = pca.fit_transform(df[genes].values)
    df["PC1"] = pcs[:, 0]
    df["PC2"] = pcs[:, 1]
    plt.figure(figsize=(8,6))
    sns.scatterplot(
        x="PC1", y="PC2",
        hue="cancer_type", style="cluster",
        data=df, s=70, alpha=0.85, palette="tab10"
    )
    plt.title(title + f" — PC1 {pca.explained_variance_ratio_[0]*100:.1f}% | PC2 {p
    plt.tight_layout()
    plt.savefig(out_png)
    plt.show()

def predict_cluster(new_values, scaler, kmeans, genes, meta_sub, clusters, expr_sub
    """
    Predict cluster for a new VEGF vector [VEGFA, VEGFB, VEGFC] (log2 TPM),
    and report the most common cancer types in that cluster (from subset refit).
    """
    sample_df = pd.DataFrame([new_values], columns=genes)
    x_std = scaler.transform(sample_df.values)
    c = int(kmeans.predict(x_std)[0])
    subset_meta = meta_sub.copy()
    subset_meta["cluster"] = pd.Series(clusters, index=expr_sub_columns).values
    in_c = subset_meta[subset_meta["cluster"] == c]
    top = in_c["cancer_type"].value_counts().head(3)
    print("\n--- Prediction Result ---")
    print(f"Input ({genes}): {dict(zip(genes, new_values))}")
    print(f"Predicted cluster: {c}")
    if not top.empty:
        print("Most common cancer types in this cluster:", ", ".join(top.index))
    else:
        print("Cluster composition unavailable.")
    return c

# ==========================
# VALIDATED 3-CANCER PIPELINE
# ==========================
def run_subset_kmeans_validation(expr, meta, test_size=0.5, random_state=42):
    """
    Steps:
      1) Select the 3 most separated cancers in VEGF-PCA space (min 15 samples each
      2) Subset to those cancers only
      3) 50/50 stratified split (train/test)
      4) Scale on TRAIN only
      5) Choose k by TRAIN silhouette over k=3..6 (enforce ≥3)
      6) Train KMeans on TRAIN; predict TEST; report:
         - Train/Test silhouette
         - Test ARI (clusters vs cancer_type)
         - Test accuracy (%) via train-majority mapping; confusion matrix
      7) Refit on ALL subset for composition plot, focused PCA, and ANOVA
    Returns dict with results and fitted scaler/kmeans for prediction.
    """
    # 1) pick farthest 3 cancer types
    pca_df = _pca_on_vegf_all(expr, meta, random_state=random_state)
    chosen = _select_farthest_3_cancers(pca_df, min_per_type=15)
    print(f"[Selection] 3 farthest cancer types on VEGF-PCA: {chosen}")

    # 2) subset
    keep_samples = pca_df.loc[pca_df["cancer_type"].isin(chosen), "sample"].values
    expr_sub = expr.loc[GENES, keep_samples]
    meta_sub = meta[meta["sample"].isin(keep_samples)].drop_duplicates("sample").co

    # features/labels
    X_sub = expr_sub.T.values
    y_types = meta_sub.set_index("sample").loc[expr_sub.columns, "cancer_type"].val

    # 3) split
    X_tr, X_te, y_tr, y_te = train_test_split(
        X_sub, y_types, test_size=test_size, random_state=random_state, stratify=y_
    )
    # 4) scale on train only
    scaler_tr = StandardScaler().fit(X_tr)
    X_tr_std = scaler_tr.transform(X_tr)
    X_te_std = scaler_tr.transform(X_te)

    # 5) choose k on train only (k>=3)
    k_candidates = range(3, 7)
    sil_train = {}
    best_k, best_sil = None, -1.0
    for k in k_candidates:
        km = KMeans(n_clusters=k, n_init=50, random_state=random_state)
        tr_labels_tmp = km.fit_predict(X_tr_std)
        sil = silhouette_score(X_tr_std, tr_labels_tmp)
        sil_train[k] = sil
        if sil > best_sil:
            best_sil = sil
            best_k = k
    print(f"[Model selection] Train silhouette by k = {sil_train} -> best_k = {best

    # train final & silhouette plots (train/test)
    km_final = KMeans(n_clusters=best_k, n_init=50, random_state=random_state)
    tr_labels = km_final.fit_predict(X_tr_std)
    _plot_silhouette(X_tr_std, tr_labels,
                     title=f"Silhouette (TRAIN) k={best_k}",
                     out_png=os.path.join(RESULTS_DIR, "silhouette_train.png"))
    te_labels = km_final.predict(X_te_std)
    _plot_silhouette(X_te_std, te_labels,
                     title=f"Silhouette (TEST) k={best_k}",
                     out_png=os.path.join(RESULTS_DIR, "silhouette_test.png"))

    # metrics
    train_sil = silhouette_score(X_tr_std, tr_labels)
    test_sil  = silhouette_score(X_te_std, te_labels)
    ari_test  = adjusted_rand_score(y_te, te_labels)

    # majority mapping from train clusters -> labels; compute test accuracy
    cluster_to_label = _majority_label_map(tr_labels, y_tr)
    y_pred_test = np.array([cluster_to_label[int(c)] for c in te_labels])
    test_acc = (y_pred_test == y_te).mean()

    classes = sorted(chosen)
    cm = confusion_matrix(y_te, y_pred_test, labels=classes)
    cm_df = pd.DataFrame(cm, index=[f"Actual_{c}" for c in classes],
                            columns=[f"Pred_{c}" for c in classes])
    print(f"Train silhouette={train_sil:.3f} | Test silhouette={test_sil:.3f} | "
          f"Test ARI={ari_test:.3f} | Test accuracy={100*test_acc:.1f}%")
    print("\n[Test confusion matrix]\n", cm_df)
    cm_df.to_csv(os.path.join(RESULTS_DIR, "subset_test_confusion_matrix.csv"))

    # 7) refit on ALL subset for plots/stats
    scaler_all = StandardScaler().fit(X_sub)
    X_all_std = scaler_all.transform(X_sub)
    km_all = KMeans(n_clusters=best_k, n_init=50, random_state=random_state)
    all_clusters = km_all.fit_predict(X_all_std)

    # cluster composition
    comp_df = meta_sub.copy()
    comp_df["cluster"] = pd.Series(all_clusters, index=expr_sub.columns).values
    plt.figure(figsize=(7,5))
    sns.countplot(x="cluster", hue="cancer_type", data=comp_df, palette="tab20")
    plt.title(f"Subset Cluster Composition (k={best_k})")
    plt.tight_layout()
    plt.savefig(os.path.join(RESULTS_DIR, "subset_cluster_composition.png"))
    plt.show()

    # focused PCA scatter (3 cancers)
    _focused_pca_scatter(
        expr_sub=expr_sub, meta_sub=meta_sub, clusters=all_clusters, genes=GENES,
        title="Focused PCA (3 cancers)",
        out_png=os.path.join(RESULTS_DIR, "focused_pca_clusters.png")
    )

    # ANOVA across clusters (all subset samples)
    df_all = pd.DataFrame(X_sub, index=expr_sub.columns, columns=GENES)
    df_all["cluster"] = all_clusters
    anova_rows = []
    for g in GENES:
        arrays = [df_all.loc[df_all["cluster"] == c, g].values for c in np.unique(a
        F, p = stats.f_oneway(*arrays)
        anova_rows.append({"gene": g, "F": float(F), "p": float(p)})
    anova_df = pd.DataFrame(anova_rows)
    print("\n ANOVA across clusters (subset, all data):\n", anova_df)
    anova_df.to_csv(os.path.join(RESULTS_DIR, "subset_anova_clusters.csv"), index=F

    per_cluster = df_all.groupby("cluster")[GENES].agg(["mean","std","count"])
    per_cluster.to_csv(os.path.join(RESULTS_DIR, "subset_per_cluster_gene_summary.c

    return {
        "chosen_types": chosen,
        "silhouette_train_by_k": sil_train,
        "best_k": int(best_k),
        "train_silhouette": float(train_sil),
        "test_silhouette": float(test_sil),
        "test_ari": float(ari_test),
        "test_accuracy": float(test_acc),
        "test_confusion_matrix": cm_df,
        "cluster_to_label_map": cluster_to_label,
        "anova_subset_all": anova_df,
        "per_cluster_summary": per_cluster,
        "scaler_all": scaler_all,
        "kmeans_all": km_all,
        "expr_sub": expr_sub,
        "meta_sub": meta_sub,
        "all_clusters": all_clusters
    }

# ==========================
# MAIN
# ==========================
def main():
    expr, meta = load_data(EXPR_FILE, META_FILE)
    # Descriptive exploration on ALL cancers
    compute_statistics(expr, meta)
    plot_expression_by_cancer_type(expr, meta)
    plot_pca(expr, meta)
    plot_cluster_composition(expr, meta)  # exploratory only

    # Validated 3-cancer pipeline (50/50 split)
    out = run_subset_kmeans_validation(expr, meta, test_size=0.5, random_state=42)




if __name__ == "__main__":
    main()
```

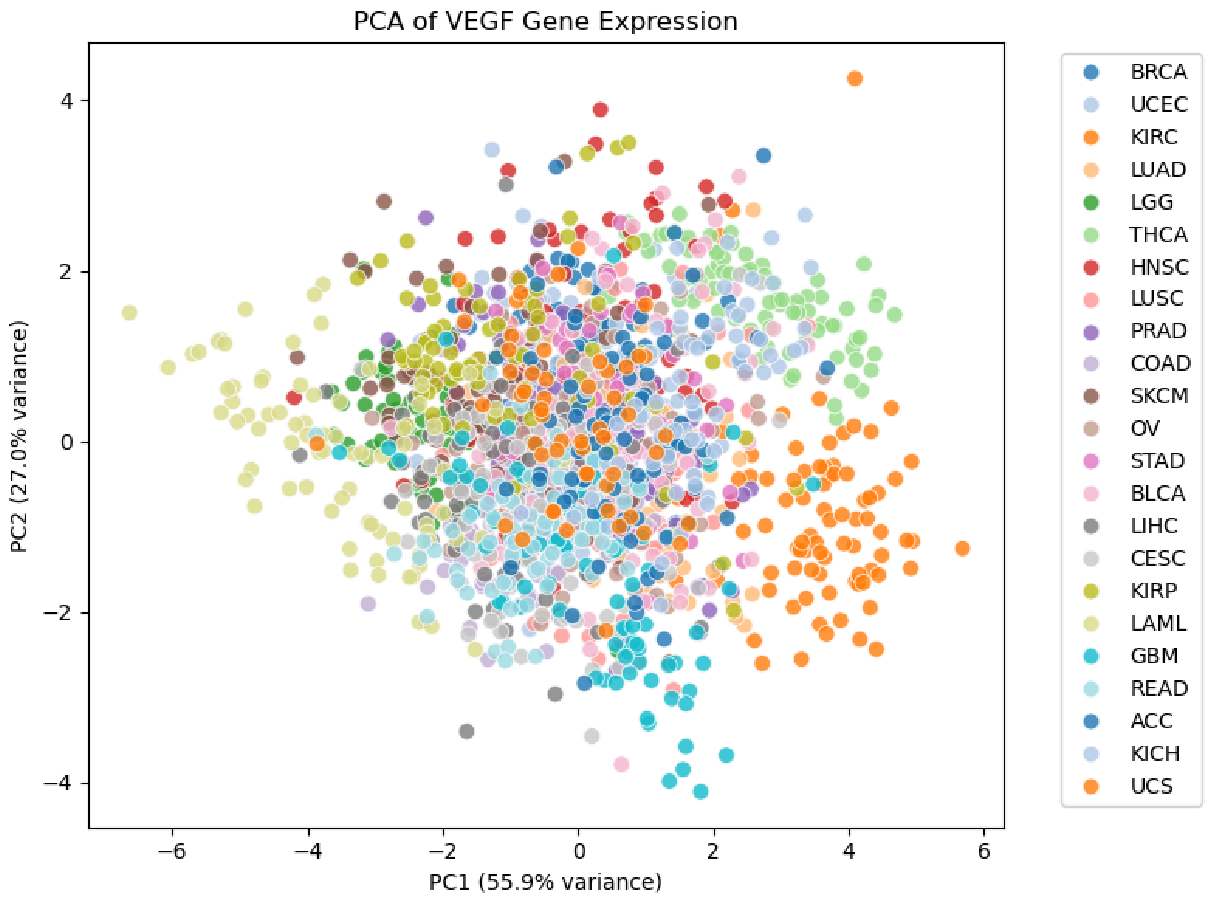
```
VEGFA Expression Summary (mean ± std, count), sorted by mean:

                 mean       std  count
cancer_type
KIRC         9.886018  1.122678     80
GBM          7.742262  1.550111     80
THCA         7.700020  1.068883     80
KICH         7.430506  0.770311     66
LUAD         7.405633  1.104648     80
ACC          7.129720  0.932957     79
BLCA         7.082584  1.150286     80
LUSC         6.975820  0.982748     80
CESC         6.838443  1.064617     80
READ         6.794130  0.732453     80
OV           6.722641  1.017300     80
STAD         6.606184  0.927776     80
PRAD         6.601118  1.472526     80
COAD         6.543043  0.672364     80
UCS          6.469420  1.179160     57
UCEC         6.301814  1.019521     80
BRCA         6.277238  1.011875     80
LIHC         6.142740  1.062468     80
HNSC         6.015108  1.170085     80
SKCM         5.374195  1.331442     80
KIRP         5.274447  1.436511     80
LGG          4.863521  1.002417     80
LAML         4.057543  1.466749     80

VEGFA ANOVA F=83.04, p=1.233e-253

VEGFB Expression Summary (mean ± std, count), sorted by mean:

                 mean       std  count
cancer_type
KICH         8.734285  0.599346     66
KIRP         7.806962  0.498744     80
KIRC         7.639225  0.606599     80
SKCM         7.523584  0.691904     80
GBM          7.318562  0.396535     80
LGG          7.256720  0.509887     80
UCS          7.188050  0.621050     57
OV           7.152939  0.516797     80
ACC          6.905926  0.671422     79
UCEC         6.902252  0.522903     80
THCA         6.802874  0.459878     80
PRAD         6.802169  0.454012     80
LUAD         6.768293  0.637091     80
BRCA         6.726119  0.575330     80
LUSC         6.582976  0.565045     80
STAD         6.446149  0.649025     80
CESC         6.223652  0.544856     80
BLCA         6.210119  0.821004     80
HNSC         6.201772  0.589906     80
READ         6.047814  0.723163     80
COAD         5.891562  0.770427     80
LAML         5.094861  0.550005     80
```

```
LIHC         5.085717  1.626412     80

VEGFB ANOVA F=114.05, p=5.909e-320

VEGFC Expression Summary (mean ± std, count), sorted by mean:

                 mean       std  count
cancer_type
THCA         6.145960  0.480932     80
KIRC         4.341789  0.889432     80
KICH         4.124832  1.235474     66
HNSC         3.853435  1.448494     80
BLCA         3.676909  1.344303     80
BRCA         3.614277  0.711051     80
ACC          3.508590  1.152482     79
STAD         3.499386  0.756852     80
UCEC         3.227999  1.042867     80
LUAD         3.193095  1.029295     80
UCS          3.141796  1.184754     57
PRAD         2.985127  0.690835     80
LUSC         2.915423  1.158114     80
KIRP         2.867487  1.040963     80
OV           2.839913  1.133232     80
SKCM         2.546649  1.081491     80
LIHC         2.360408  0.912410     80
CESC         2.294583  1.152805     80
COAD         2.026583  0.792095     80
READ         1.990320  0.846577     80
GBM          1.732651  0.892735     80
LGG          1.574082  0.655929     80
LAML         1.010846  0.708218     80

VEGFC ANOVA F=94.12, p=1.188e-278
```

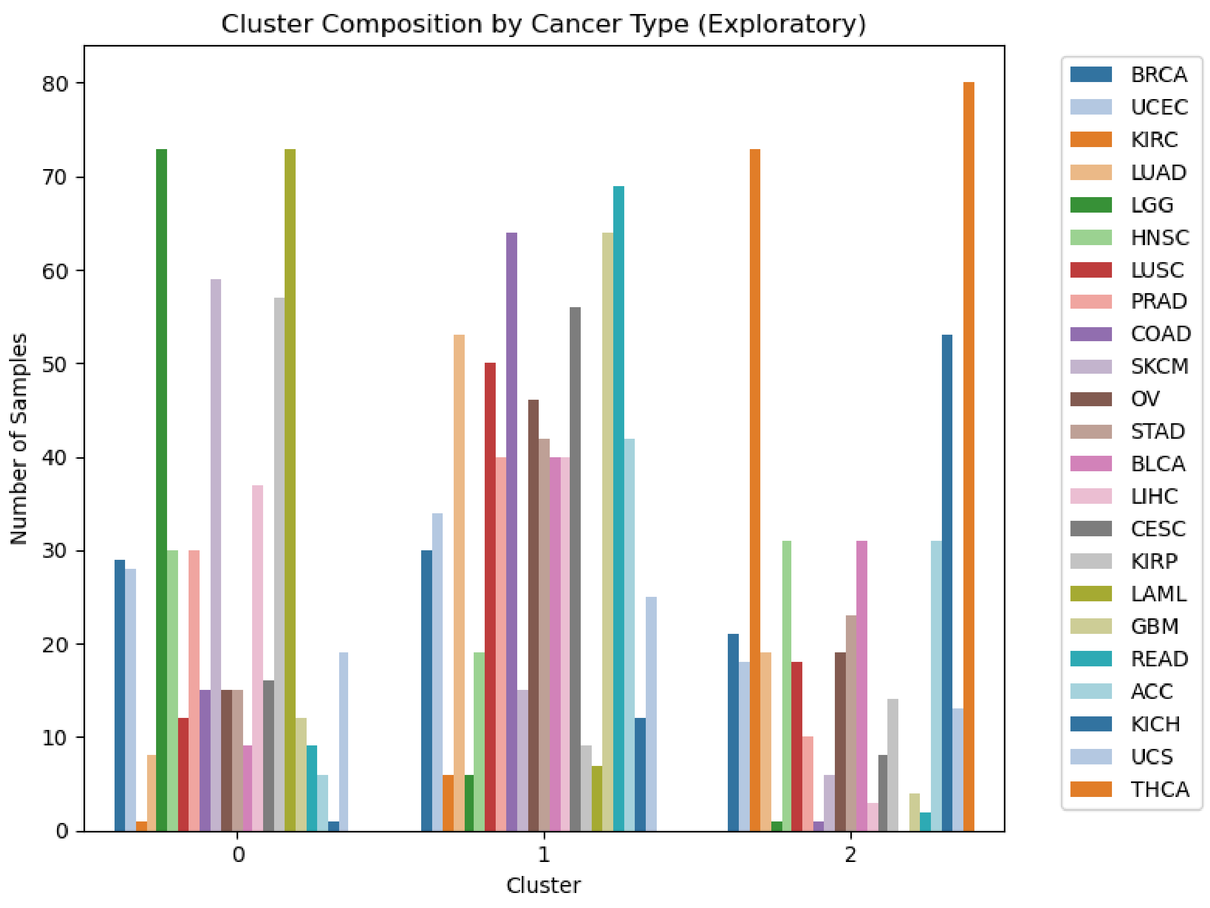



VEGFB Expression by Cancer Type
VEGFB
cancer_type
BRCA UCEC KIRC LUAD LGG THCA HNSC LUSC PRAD COAD SKCM OV STAD BLCA LIHC CESC KIRP LAML GBM READ ACC KICH UCS


VEGFC Expression by Cancer Type
VEGFC
cancer_type
BRCA UCEC KIRC LUAD LGG THCA HNSC LUSC PRAD COAD SKCM OV STAD BLCA LIHC CESC KIRP LAML GBM READ ACC KICH UCS

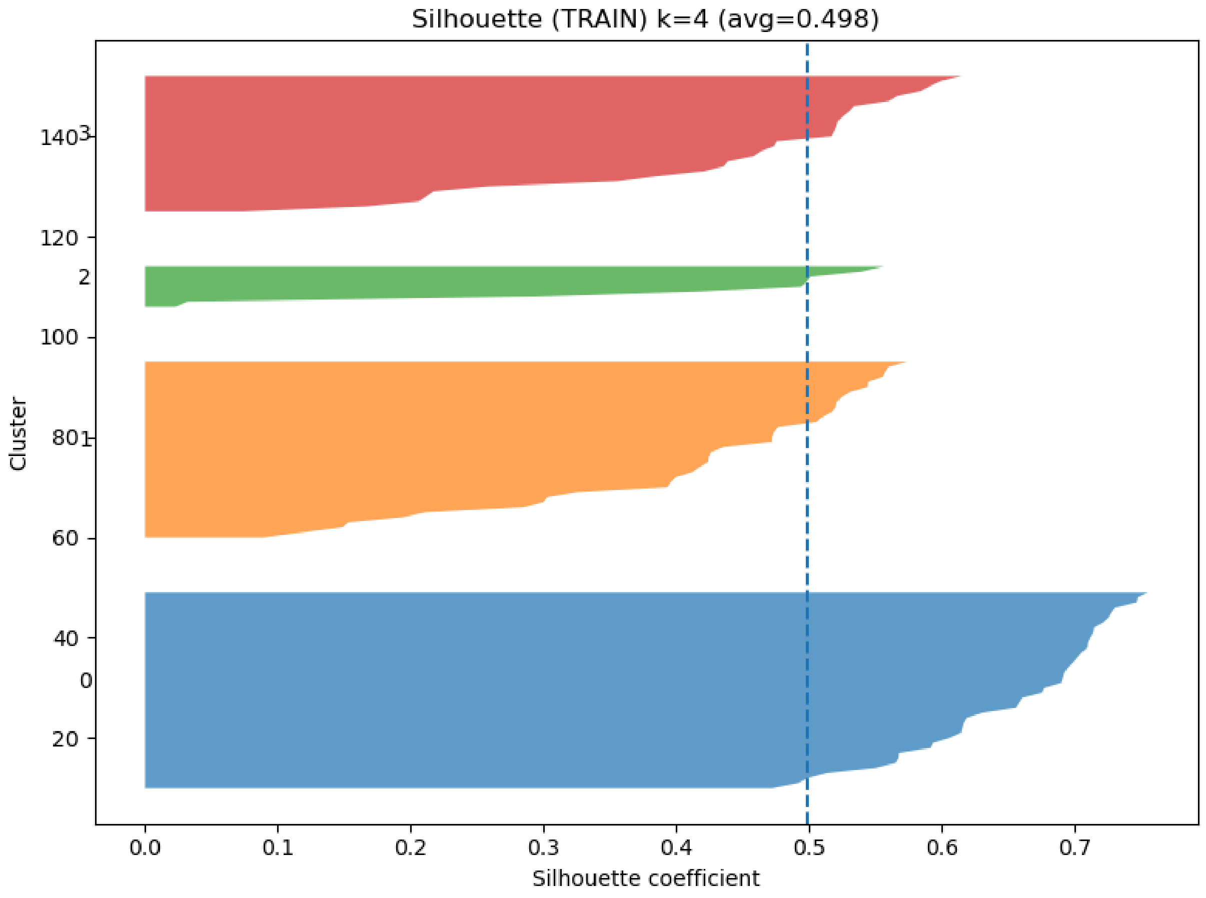


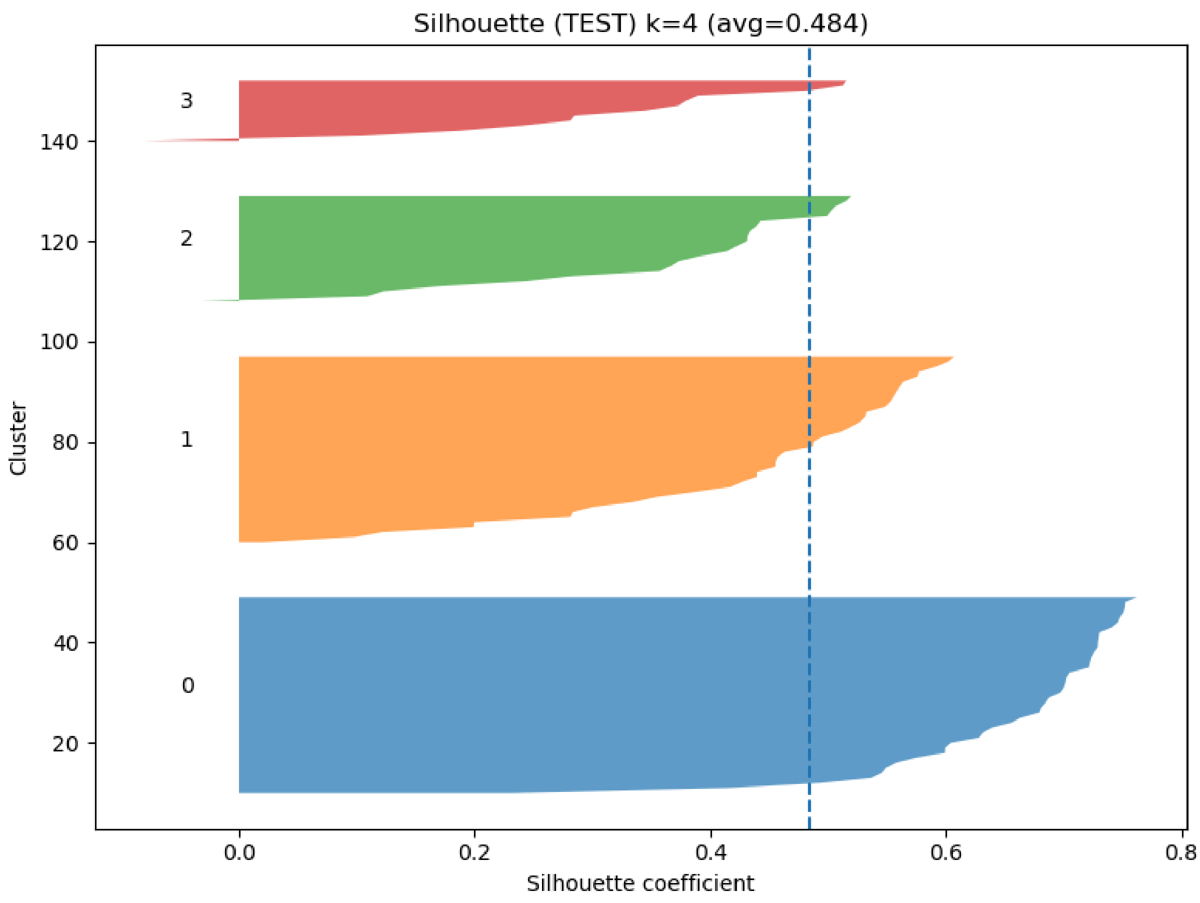
```
c:\Users\danig\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=8.
  warnings.warn(
```



```
[Selection] 3 farthest cancer types on VEGF-PCA: ['KICH', 'KIRC', 'LAML']
c:\Users\danig\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\danig\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\danig\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\danig\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
[Model selection] Train silhouette by k = {3: np.float64(0.49619985638392017), 4: n
p.float64(0.4983640008354871), 5: np.float64(0.38340212009559843), 6: np.float64(0.3
879612982550106)} -> best_k = 4
```

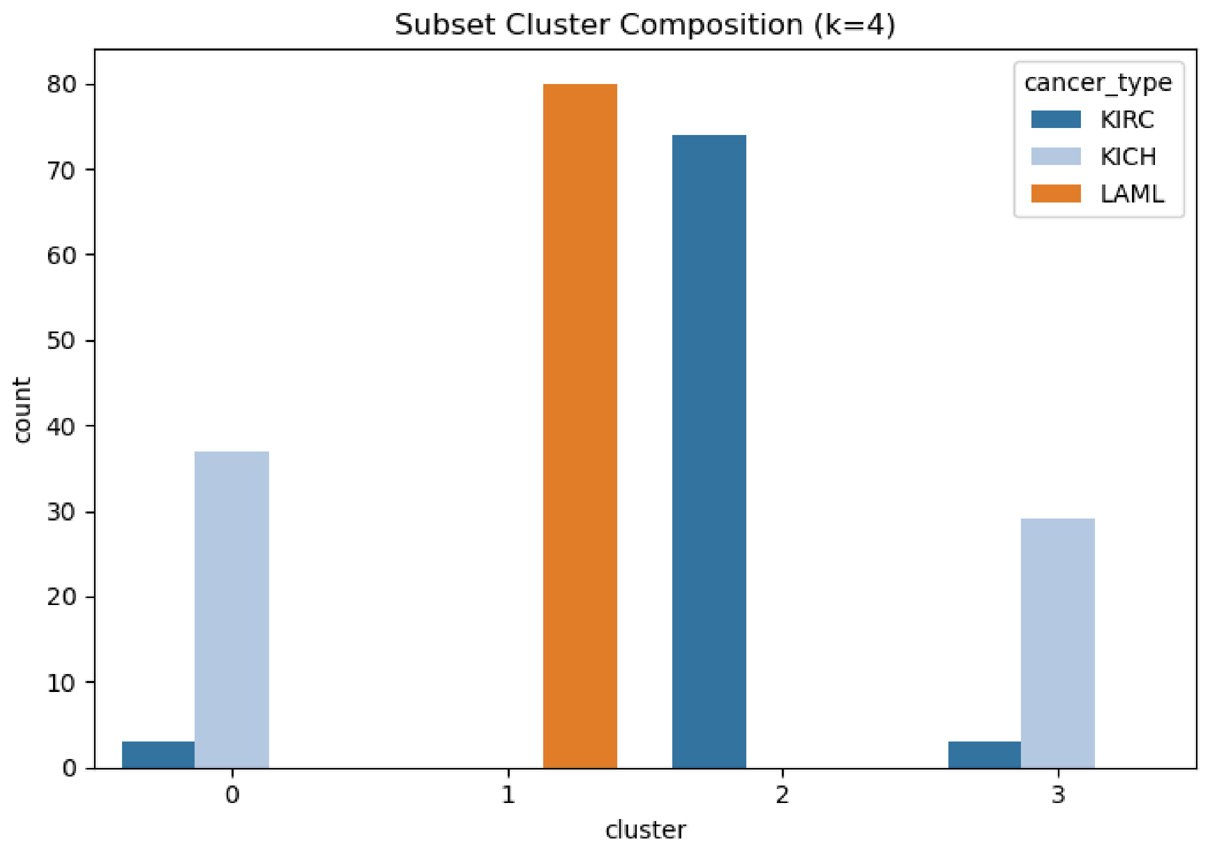
```
c:\Users\danig\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```

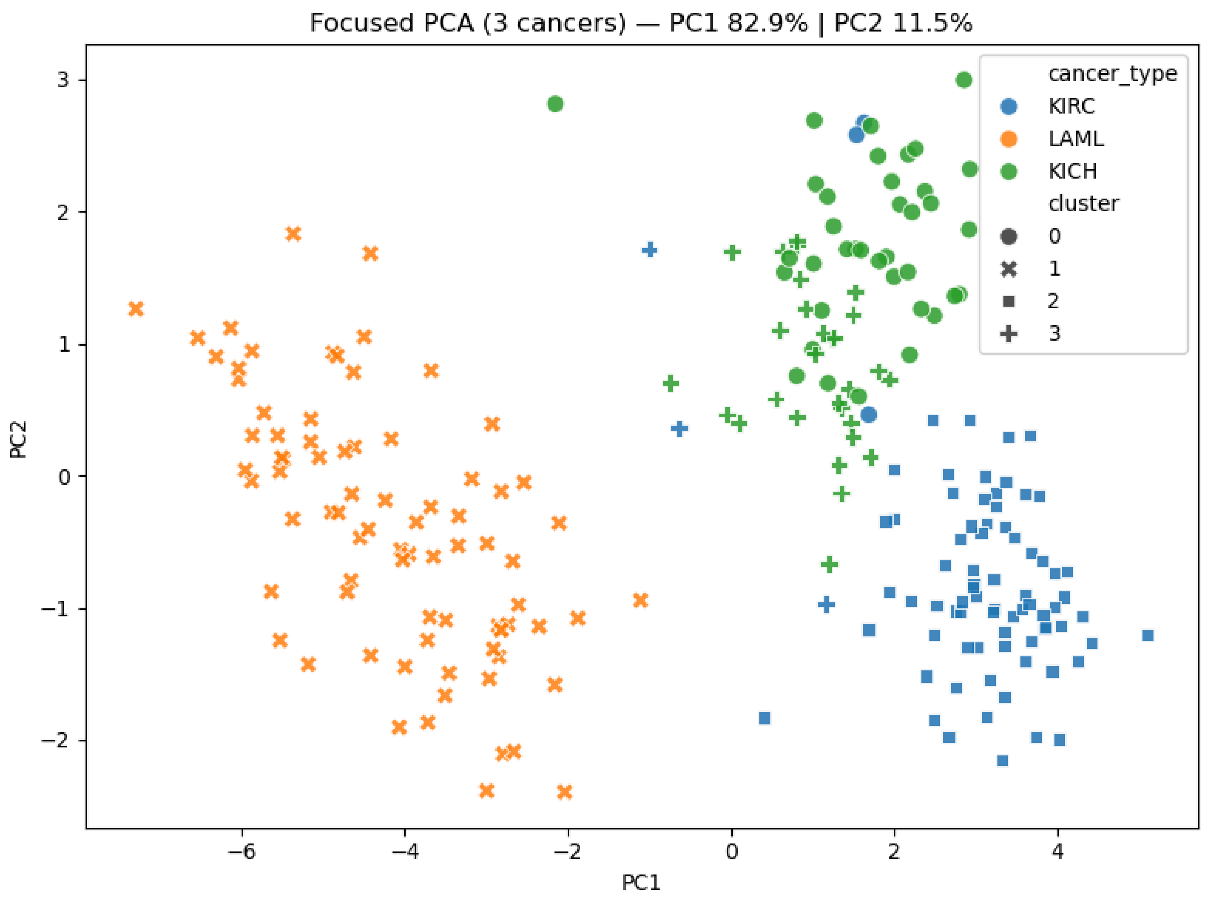




```
Train silhouette=0.498 | Test silhouette=0.484 | Test ARI=0.866 | Test accuracy=98.
2%

[Test confusion matrix]
             Pred_KICH  Pred_KIRC  Pred_LAML
Actual_KICH         33          0          0
Actual_KIRC          2         38          0
Actual_LAML          0          0         40
```

```
c:\Users\danig\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarn
ing: KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```


Subset Cluster Composition (k=4)
cancer_type
KIRC
KICH
LAML
count
cluster
0
1
2
3
0
10
20
30
40
50
60
70
80


Focused PCA (3 cancers) — PC1 82.9% | PC2 11.5%
cancer_type
KIRC
LAML
KICH
cluster
0
1
2
3
PC1
PC2
−6
−4
−2
0
2
4
−2
−1
0
1
2
3

```
 ANOVA across clusters (subset, all data):
     gene           F             p
0  VEGFA  413.726439  1.375137e-90
1  VEGFB  504.924187  7.601132e-99
2  VEGFC  389.654611  3.769745e-88
```

## Verify and validate your analysis:

ARI scores the similarity of pairs within the predicted clustering set alongside the actual known class. "Similarity" is defined by whether the pair is seen in the same cluster as the actual data. Then, the score is adjusted to 1 for full agreement and 0 for randomized or incorrect. The given ARI from the predictions was 0.866, which is closer to 1, supporting the method's prediction through clustering. Furthermore, to validate the unsupervised model, it is best to use a silhouette score. This is a single number that quantifies the clusters/points and how close they are to each other without any true labels. The values range from +1 to -1, where the higher the value, the more distinct the points are. The train silhouette score is 0.498 (based on the top 50% of data), while the test score is 0.484 (all data points) for the top three cancers for VEGF. Since the test is positive and near each other, the clustering is fairly stable and not invalid. The difference between this score and ARI is that the two trained values (train vs. test) are a comparison of the clustered data between each other.  In summary, ARI validates the prediction of the trained model, while the silhouette score compares the clustering k-means model by only internal metrics.

Primarily, two of the three cancers that were isolated to be clustered would be kidney renal cell carcinoma (KIRC) and kidney chromophobe renal cell carcinoma (KICH). While these cancers may be distinct in their origins, genetics, and treatment, it is still similar in location and spread. It is important to determine the carcinoma early on to prevent spread within the organ, to the second organ, or further throughout the body. It is reasonable to see overlap within the KIRC and KICH while being able to predict the VEGF properties separately from the trained model. Both cancers are seen to be fast-spreading, angiogenesis-impacted diseases. Furthermore, the two failed predictions from the trained model were predicting KICH for KIRC, which is expected by the similarities.

The second major cluster is based on acute myeloid leukemia. Based on articles from 2000 to current research, the prevalence of VEGF family excitation is common. This cancer type is typically characterized by the rapid growth of white blood cells called myeloblasts. In further research, it is seen that this excessive growth of myeloblasts is heavily related to the VEGA and VEGB genetic biomarkers, which differ from the graphs above. This may be because myeloblast expression comes off as a different marker from angiogenesis than solely the VEGF family, which is what distinguishes itself so well in the clustering.

https://www.nature.com/articles/nrurol.2010.46.pdf

https://www.moffitt.org/cancers/kidney-renal-cell-cancer/diagnosis/types/

https://pmc.ncbi.nlm.nih.gov/articles/PMC11113417/

https://www.tandfonline.com/doi/pdf/10.1179/1607845413Y.0000000139

## Conclusions and Ethical Implications:

This project had a focus on highlighting angiogenesis through VEGF biomarkers in cancers and comparing them with one another to distinguish prevalence. To better understand angiogenesis, we analyzed a large set of data with information about patients who face cancer. Our code highlighted the VEGA, VEGFB, and VEGFC expression and ranked all cancers in comparison. Based on only VEGFA, KIRC seemed to have been most impacted by angiogenesis. Utilizing this, along with the underexpressed LAML within the VEGF biomarkers, k-means clustering was trained as an unsupervised model in predicting angiogenesis within cancers. This model answers the question: are cancers differentiable by only VEGF expression, as some may be impacted by angiogenesis more than others? The overlap is excessive and shows that all cancers may be impacted by angiogenesis, but cannot be simply differentiated from this biomarker. However, to further train the unsupervised model, we selected three of the most differentiated cancers to see if prediction is possible on a smaller data set. As shown in the verification above, the prediction model purely trained on internal metrics was able to separate cancers typically induced by angiogenesis from cancers that spread in other ways.

The analysis surrounded comparison of cancers and visualization of expression of certain biomarkers. To do so, the code surrounded simple python matplots and statistics. The only ethical implication which is commonly discussed is unsupervised modeling usage and k-means clustering. In larger data sets or applications k-means clustering uses internal metrics and does not serapate information as needed. Privacy issues come into play and if more features from the large class was given to the code, it may reveal information or need private information of the patients to better its strength in prediction. Furthermore, bias was used in the code. While the application of predicting cancers can grow to more features, prior to coding the k-means cluster we purposely selected the most distinct points from the PCA of all cancers. K-means clustering may push for more bias in unsupervised learning for "more valid" results. Both of these factors are major ethical considerations for larger use of this k-means clustering model.

## Limitations and Future Work:

Increasing features, changing classification threshold, and finally further validate the set with unseen test data. Spectral clustering, GMMM

## NOTES FROM YOUR TEAM:

Need to add train-test split and silhouettes score+visualization

## QUESTIONS FOR YOUR TA:

*These are questions we have for our TA.*