END-TO-END ANALYTICS FOR CANCER MORTALITY

In [349…

```
#All imports

import pandas as pd
from snowflake.snowpark import Session
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt

from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.multivariate.pca import PCA

from sklearn.preprocessing import  OneHotEncoder, MinMaxScaler, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.inspection import permutation_importance
from sklearn.model_selection import learning_curve

pd.set_option("display.max_rows", 10)
```

Data Import

Format File -> Variable

BLS regions -> regions

Cancer deaths -> deaths

Socail determinants -> determinants

State demographics -> demographics

In [350…

```
deaths = pd.read_csv("./cancer_deaths.csv")
regions = pd.read_csv("./bls_regions.csv")
determinants = pd.read_csv("./social_determinants.csv")
demographics = pd.read_csv("./state_demographics.csv")
```

DATA Wrangling/Cleaning

Converted deaths per 100,000 to death per 1000 (keeping a standard scale)

In [351…

```
#since we have deaths rate per 100,000  converting this data to  deaths per 1000
numerical_column_deaths = deaths.select_dtypes(include="number").columns
deaths[numerical_column_deaths] = deaths[numerical_column_deaths] * 1000 / 10000
```

```
#Displaying deths per 1000
deaths.head()
```

Out[351…

	State	Cancer Death Rate.Total	Cancer Death Rate.Breast	Cancer Death Rate.Colorectal	Cancer Death Rate.Lung
0	Alabama	2.142	0.274	0.194	0.664
1	Alaska	1.281	0.178	0.119	0.366
2	Arizona	1.656	0.233	0.149	0.423
3	Arkansas	2.239	0.279	0.212	0.733
4	California	1.509	0.230	0.140	0.345

Pivoted Social determinant data to columns for clubbing with other cancer data and better analysis

In [352…

```
# We need to pivot social determinant as all the data is saved in columns and we

pivot_df_determinant = determinants.pivot_table(
    index = "State",
    columns = "Variable",
    values = "Count",
    aggfunc="mean"
).fillna(0)
pivot_df_determinant.head()
```

Out[352…

Variable	Cigarette Use.Adult.Smokers	Cigarette Use.Adult.Tried to Quit	Social Determinants.Consumed 1 or More Fruits per Day	Determinants.Co 1 or More Ve
State				
Alabama	590423.0	426231.0	2723159.0	3
Alaska	82005.0	53397.0	424633.0	
Arizona	661867.0	370069.0	4176477.0	5
Arkansas	388531.0	217744.0	1644292.0	2
California	2644835.0	1459787.0	25225386.0	30

Merged all the data state wise keeping same scale population/1000

In [353…

```
#Since we already converted deaths per 1000 now we are merging other data state
cancer_df_merged = pd.merge(regions, pivot_df_determinant, on="State" ,how="inne
cancer_df_merged = pd.merge(cancer_df_merged, demographics, on="State" ,how="inn

cancer_data_numerical_columns = cancer_df_merged.select_dtypes(include="number")

#Converting each column value to rate per 1000
for column in cancer_data_numerical_columns:
    if column != 'State Demographics.Population':
        cancer_df_merged[column] = (cancer_df_merged[column] * 1000 )/ cancer_df

#Merging cancer deaths and rest of the cancer data
cancer_df_merged = pd.merge(cancer_df_merged, deaths, on="State" ,how="inner")
cancer_df_merged.head()
```

Out[353…

	State	Region	Cigarette Use.Adult.Smokers	Cigarette Use.Adult.Tried to Quit	Social Determinants.Consumed 1 or More Fruits per Day	D
0	Alabama	Southeast	117.513976	84.834262	541.999957	
1	Alaska	West	111.816207	72.808366	578.999470	
2	Arizona	West	92.549369	51.747032	583.999977	
3	Arkansas	Southwest	129.014745	72.303591	545.999965	
4	California	West	66.893118	36.920906	637.999993	

5 rows × 31 columns

Snowflake Connection and data upload

Now here we are pushing the data to Snowflake to do analysis using SQL and to further do EDA on power BI.

In [354…

```
# Write Pandas DataFrame to Snowflake table (auto-creates schema)
#Creating session config
conn = {
"user" : "SHIPATEL302",
"password" : "Patelshivani@123",
"account" :  "GZHMWWI-YMC90175",
"role" : "ACCOUNTADMIN",
"warehouse" : "SNOWFLAKE_LEARNING_WH",
"database" : "CANCER",
"schema" : "PUBLIC"
}

# Create session
session = Session.builder.configs(conn).create()
#Write to table
session.write_pandas(cancer_df_merged, 'cancer_deaths_complete', auto_create_tab
```

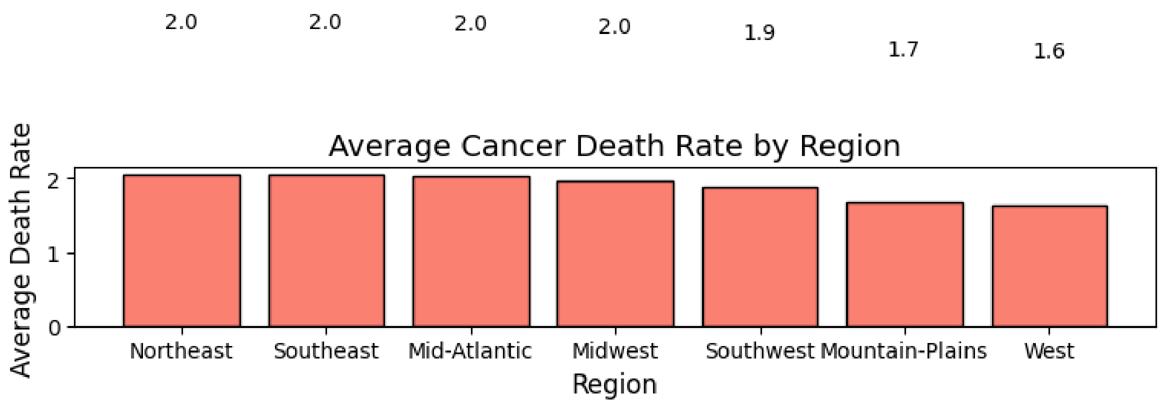
Out[354…

```
<snowflake.snowpark.table.Table at 0x2b27ea14b90>
```

Average death by region plot

In [355…

```
 #Group by Region and calculate average
region_avg = cancer_df_merged.groupby("Region")["Cancer Death Rate.Total"].mean(

# Sort descending by death rate
region_avg = region_avg.sort_values(by="Cancer Death Rate.Total", ascending=Fals
# Plot
plt.figure(figsize=(8,6))
plt.bar(region_avg["Region"], region_avg["Cancer Death Rate.Total"], color="salm

plt.title("Average Cancer Death Rate by Region", fontsize=14)
plt.xlabel("Region", fontsize=12)
plt.ylabel("Average Death Rate", fontsize=12)

# Add labels on bars
for i, val in enumerate(region_avg["Cancer Death Rate.Total"]):
    plt.text(i, val + 2, round(val, 1), ha="center", fontsize=10)

plt.tight_layout()
plt.show()
```

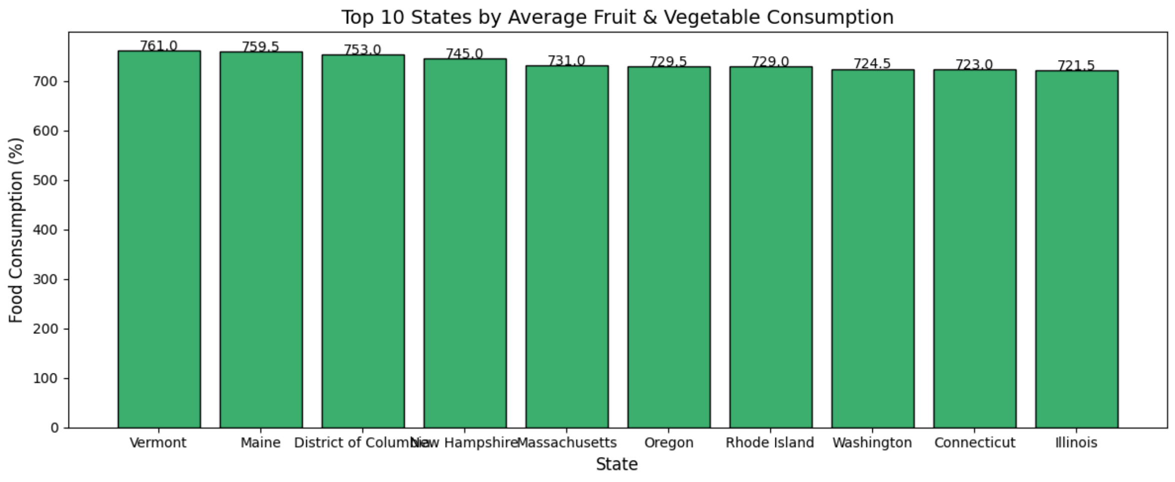


Top 10 States by Average Fruit & Vegetable Consumption

In [356…

```
food_consumption = (cancer_df_merged["Social Determinants.Consumed 1 or More Fru
top10_regions = cancer_df_merged[["State", "Region"]]
top10_regions["food_consumption"] = food_consumption
top10 = top10_regions.sort_values(
    by="food_consumption", ascending=False
).head(10)
plt.figure(figsize=(12,5))
plt.bar(top10["State"], top10["food_consumption"], color="mediumseagreen", edgec

plt.title("Top 10 States by Average Fruit & Vegetable Consumption", fontsize=14)
plt.xlabel("State", fontsize=12)
plt.ylabel("Food Consumption (%)", fontsize=12)

# Show values on top of bars
for i, val in enumerate(top10["food_consumption"]):
    plt.text(i, val + 0.5, round(val, 1), ha="center", fontsize=10)

plt.tight_layout()
plt.show()
```

```
C:\Users\shivani patel\AppData\Local\Temp\ipykernel_22376\2692707606.py:3: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  top10_regions["food_consumption"] = food_consumption
```

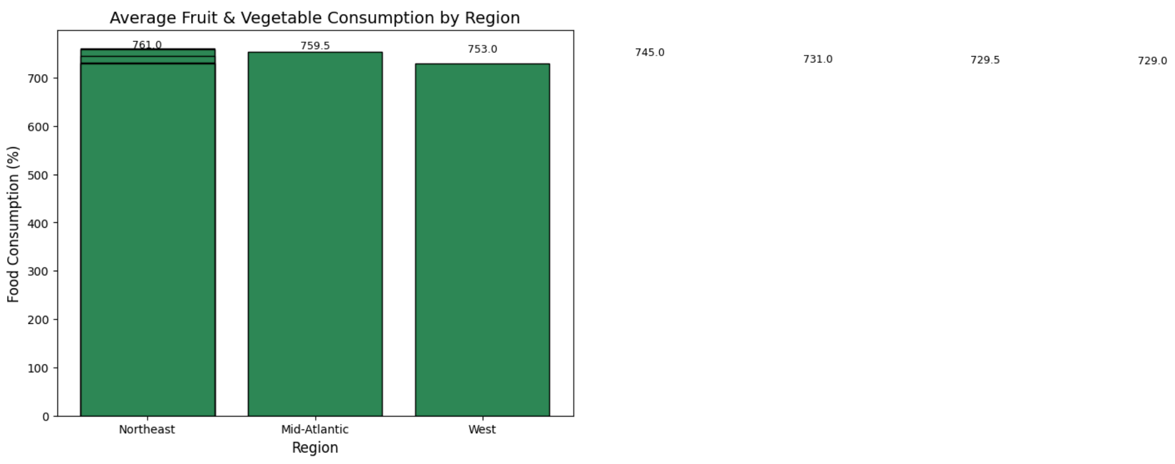


Top Average food consumption by region plot

In [357…

```
top10_regions.sort_values(
    by="food_consumption", ascending=False
).head(3)
plt.figure(figsize=(8,6))
top10 = top10.head(7)
plt.bar(top10["Region"], top10["food_consumption"], color="seagreen", edgecolor=

plt.title("Average Fruit & Vegetable Consumption by Region", fontsize=14)
plt.xlabel("Region", fontsize=12)
plt.ylabel("Food Consumption (%)", fontsize=12)

for i, val in enumerate(top10["food_consumption"]):
    plt.text(i, val + 0.5, round(val, 1), ha="center", fontsize=9)

plt.tight_layout()
plt.show()
```

```
C:\Users\shivani patel\AppData\Local\Temp\ipykernel_22376\2968305752.py:15: UserW
arning: Tight layout not applied. The left and right margins cannot be made large
enough to accommodate all Axes decorations.
  plt.tight_layout()
```



The Above EDA shows that Eastern US region is more prone to cancer death, vermont facing the largest number of deaths with cancer amongst all USA states.

Also we found that most people in northeast region eat healthy food.

We will further do statistical test to prove or findings in EDA.

Analysis Correlation map using seaborn

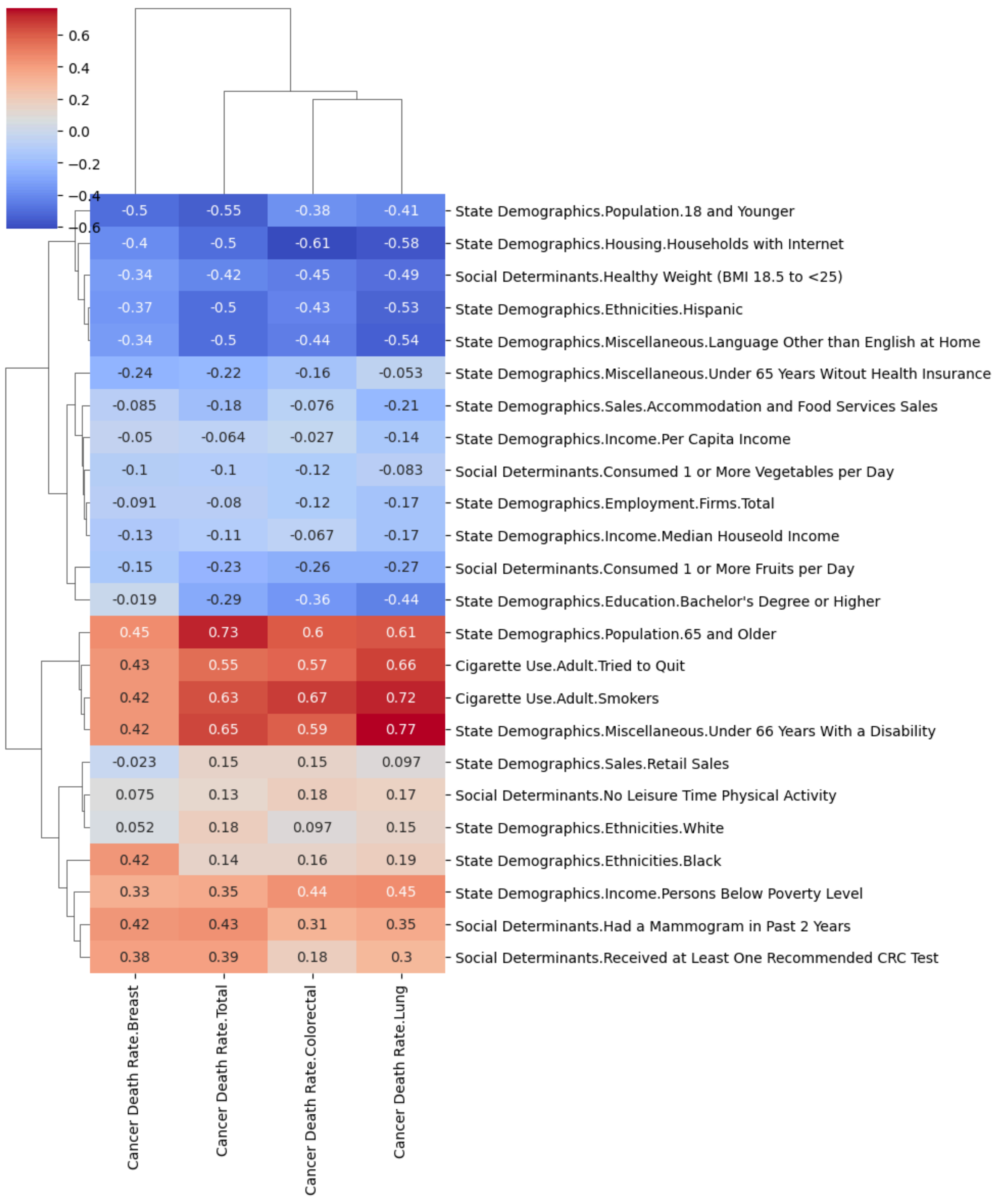
Smoking and age > 65 is highly correlated to cancer deaths
Eating fruits and vegetables are negetively correlated to cancer deaths

In [358…

```
numeric_data = cancer_df_merged.select_dtypes(include='number')
numeric_data = numeric_data.drop(columns=['State Demographics.Population'])
selected_cols = [
    "Cancer Death Rate.Total",
    "Cancer Death Rate.Breast",
    "Cancer Death Rate.Colorectal",
    "Cancer Death Rate.Lung",
]
filtered_df = cancer_df_merged[selected_cols]


# Before Revision:
# 2.2 Remove those same four variables from the rows
other_cols_df = numeric_data.drop(columns=selected_cols)



# 2.3 Correlate other variables (rows) with selected cancer columns (columns)
non_square_corr_df = pd.DataFrame()

for col in selected_cols:
    non_square_corr_df[col] = other_cols_df.corrwith(filtered_df[col])

# 2.4 : Now plot using seaborn's clustermap
sns.clustermap(non_square_corr_df, cmap="coolwarm", annot=True, figsize=(10, 12)
```

Out[358…

```
<seaborn.matrix.ClusterGrid at 0x2b274d8aa80>
```



ANOVA TEST

Performed ANOVA test to find if Region is statistically significant in causing cancer deaths

In [359…

```
# Perform an ANOVA test to see if there is a statistical significance between re

deaths = [
    cancer_df_merged.query(f"Region == '{region}'")["Cancer Death Rate.Total"]
    for region in cancer_df_merged["Region"].dropna().unique()
]


result = stats.f_oneway(*deaths)
```

```
# - The :.1f formats the F statistic to 1 decimal place
# - The :.1e formats the p-value in scientific notation with 1 decimal place
print(f"F stat: {result.statistic:.1f}, p-value: {result.pvalue:.3f}")
if (result.pvalue < 0.05):
    print(f"P value is {result.pvalue:.3f} which is way less than 0.05 therefore
```

```
F stat: 3.5, p-value: 0.006
P value is 0.006 which is way less than 0.05 therefore significant relationship i
s found between region and cancer death
```

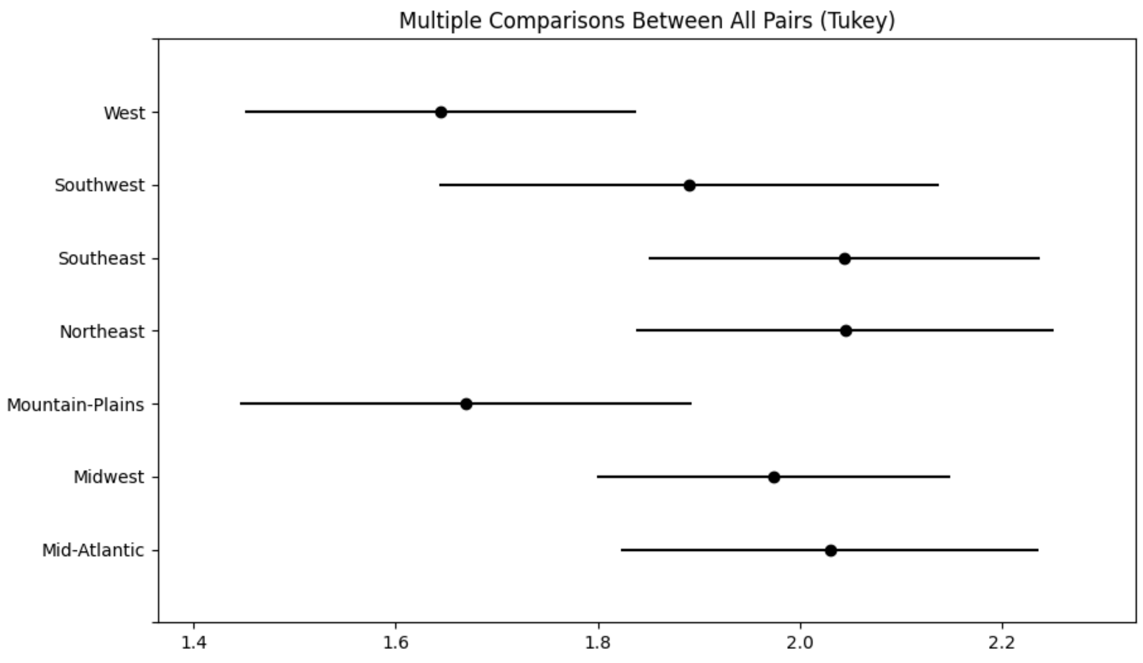
Tukey HSD Test

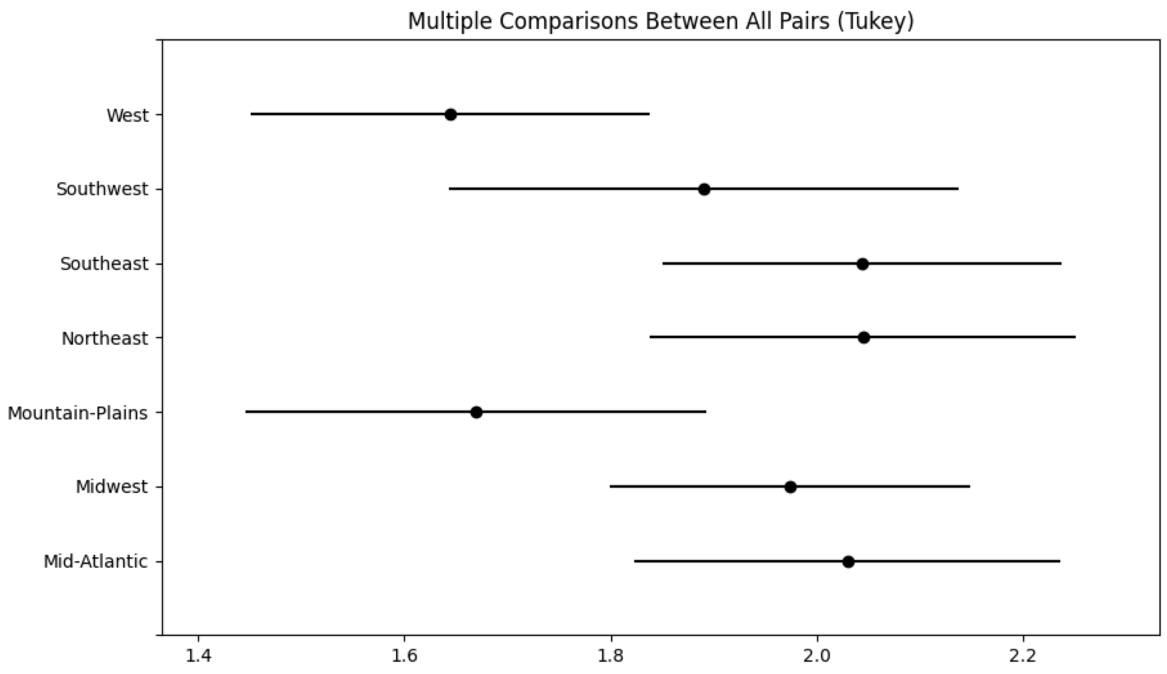
Since we found significant statistical relation between regions and deaths using ANOVA we furrther performed Tuky test to find which regions determine cancer deaths.

In [360…

```
# 1.2 Since significant difference was found between regions, perform Tukey's HS


# Create a DataFrame with cancer death rates and region
tukey_data = cancer_df_merged[["Cancer Death Rate.Total", "Region"]].dropna()

# Perform Tukey's HSD test
tukey_result = pairwise_tukeyhsd(
    endog=tukey_data["Cancer Death Rate.Total"],
    groups=tukey_data["Region"],
    alpha=0.05
)

tukey_result.plot_simultaneous()
# Plot the results
```

Out[360…





We found cancer is more prevalant in Northeast and south east regions, whereas west and mountain plains are relatevely less affected by cancer deaths.

PCA (Principal Component Analysis)

** We performed PCA converging the factors into two components:**

In [361…

```
#4.1 Drop non numeric cols
numeric_data = cancer_df_merged.select_dtypes(include='number')
numeric_data = numeric_data.drop(columns=['State Demographics.Population'])

result = PCA(

    # We're only using the numeric feature columns
    numeric_data ,
    # demean=True means that the mean will be subtracted from each feature
    # This centers the data around zero
    demean=True,
    # standardize=False means that we are NOT scaling each feature to have unit
    # If True, each feature would be divided by its standard deviation
    # R's `prcomp()` would use both center = TRUE and scale. = TRUE for standard
    standardize=True,
    # We won't get into the math behind this parameter.  Just know that setting
    # False is the more typical use case for our purposes.
    normalize=False,
    # ncomp=2 specifies that we want to keep only the first 2 principal componen
    # These will be the 2 directions that capture the most variance in the data
    ncomp=2,
)
print(result)
print(result.loadings)
```

```
Principal Component Analysis(nobs: 51, nvar: 28, transformation: Standardize (Cor
relation), normalization: False, number of components: 2, SVD)
                                                      comp_0    comp_1
Cigarette Use.Adult.Smokers                         0.300494 -0.078503
Cigarette Use.Adult.Tried to Quit                   0.290037 -0.091752
Social Determinants.Consumed 1 or More Fruits p... -0.111808  0.155776
Social Determinants.Consumed 1 or More Vegetabl... -0.028459  0.097215
Social Determinants.Had a Mammogram in Past 2 Y...  0.075903  0.279503
...                                                      ...       ...
State Demographics.Sales.Retail Sales               0.023373  0.119557
Cancer Death Rate.Total                             0.304292  0.182235
Cancer Death Rate.Breast                            0.242536  0.181728
Cancer Death Rate.Colorectal                        0.299348  0.114436
Cancer Death Rate.Lung                              0.322189  0.102431

[28 rows x 2 columns]
```

Using PCA we found that Cigrate Use and those trying to quit cigratte were major contributing factor in both components
Whereas, Eating fruits and vegetables is negetively correlated to principal components.

That means smoking and diet are key factors

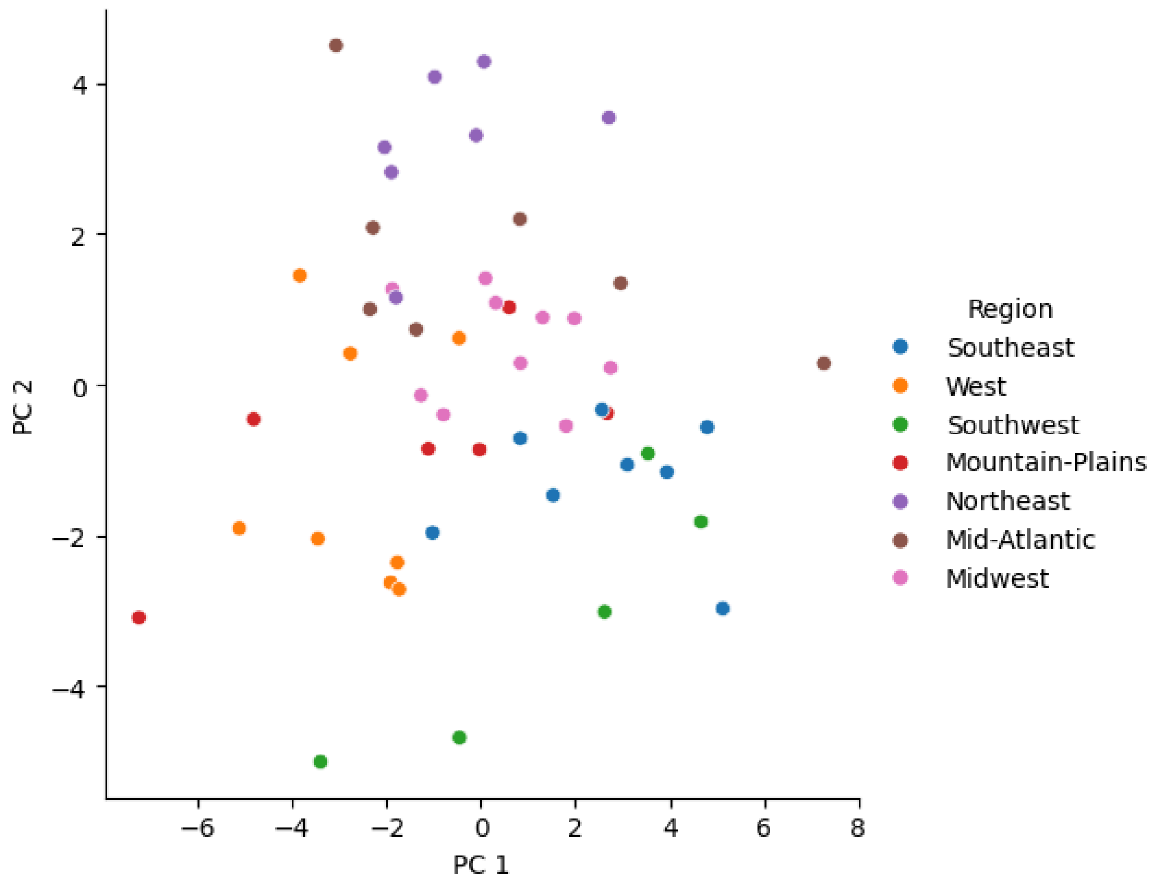
Further we plotted Region on the two components to find how different region contributes to the two principal components.

In [362…

```python
# Prepare the data for plotting
plot_data = (
    result.scores.rename(
        columns={
            # Rename the first principal component for better readability
            "comp_0": "PC 1",
            # Rename the second principal component for better readability
            "comp_1": "PC 2",
        }
    )
    # Add the Region column from the original dataset for coloring points
    .assign(Region=cancer_df_merged['Region'])
)

# Create a scatter plot using seaborn
# - relplot creates a relational plot (scatter plot in this case)
# - x and y specify which columns to use for the x and y axes
#   (the first two principal components)
# - hue colors the points based on the Species column
sns.relplot(plot_data, x="PC 1", y="PC 2", hue="Region")
```

Out[362…

```
<seaborn.axisgrid.FacetGrid at 0x2b20102f9d0>
```



Here we found South East and Northeast (blue and purple) regions are major contributors to principal component 1 & 2. Whereas, mountain plains and west (Red and orange) are least contributors to the principal components within regions.

MACHINE LEARNING

Process & Pipeline
Prepping data -> Segrigated Numerical Cols -> Split Data in Train and Test -> One hot encoding -> Normalization -> ML model Training -> Prediction Converted cancer rate to 0 and 1 to predict cancer mortality.
divided Train and test data in 80:20 ratio.
Used Logistic regression model for prediction using test data.

In [363…

```
#MACHINE LEARNING



#dropping insignificant columns
cancer_data_significant = cancer_df_merged.drop(columns=['State Demographics.Pop
#Taking out result column
result = cancer_df_merged["Cancer Death Rate.Total"]
mean_result = result.mean()
#Changing result column in category where 0= no cancer, 1= cancer:
result = (result > mean_result).astype(int)
#[0,1,1,1,0]
```

```
#dropping result columns
selected_cols = [
    "Cancer Death Rate.Total",
    "Cancer Death Rate.Breast",
    "Cancer Death Rate.Colorectal",
    "Cancer Death Rate.Lung",
]
cancer_data_significant = cancer_data_significant.drop(columns=selected_cols)
cancer_data_significant.info()
numerical_cols = cancer_data_significant.select_dtypes(include='number').columns

categorical_col = ["Region"]
#Min max scalar transforms to normalize data betwwen 0 to 1
preprocess = ColumnTransformer(
    transformers= [
        ("num", MinMaxScaler(), numerical_cols),
        ("cat", OneHotEncoder(handle_unknown="ignore"), categorical_col)

    ]
)
X_train, X_test, y_train, y_test = train_test_split(cancer_data_significant, res
#print(X_train.shape, y_train.shape)
model = Pipeline([
    ("prep", preprocess),
    ("clf", LogisticRegression(max_iter=1000))
])

model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print("actual", y_test.to_list())
print("predicted" ,y_pred)

res = 0
for i,j  in zip(y_test.to_list(), y_pred):
    if i == j:
        res += 1

print("Accuracy", res * 100/len(y_pred))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 25 columns):
 #   Column                                                                Non
-Null Count  Dtype  
---  ------                                                                ---
-----------  -----  
 0   Region                                                                51 
non-null     object 
 1   Cigarette Use.Adult.Smokers                                           51 
non-null     float64
 2   Cigarette Use.Adult.Tried to Quit                                     51 
non-null     float64
 3   Social Determinants.Consumed 1 or More Fruits per Day                 51 
non-null     float64
 4   Social Determinants.Consumed 1 or More Vegetables per Day             51 
non-null     float64
 5   Social Determinants.Had a Mammogram in Past 2 Years                   51 
non-null     float64
 6   Social Determinants.Healthy Weight (BMI 18.5 to <25)                  51 
non-null     float64
 7   Social Determinants.No Leisure Time Physical Activity                 51 
non-null     float64
 8   Social Determinants.Received at Least One Recommended CRC Test        51 
non-null     float64
 9   State Demographics.Population.65 and Older                            51 
non-null     float64
 10  State Demographics.Population.18 and Younger                          51 
non-null     float64
 11  State Demographics.Education.Bachelor's Degree or Higher              51 
non-null     float64
 12  State Demographics.Employment.Firms.Total                             51 
non-null     float64
 13  State Demographics.Ethnicities.Black                                  51 
non-null     float64
 14  State Demographics.Ethnicities.Hispanic                               51 
non-null     float64
 15  State Demographics.Ethnicities.White                                  51 
non-null     float64
 16  State Demographics.Housing.Households with Internet                   51 
non-null     float64
 17  State Demographics.Income.Median Houseold Income                      51 
non-null     float64
 18  State Demographics.Income.Per Capita Income                           51 
non-null     float64
 19  State Demographics.Income.Persons Below Poverty Level                 51 
non-null     float64
 20  State Demographics.Miscellaneous.Language Other than English at Home  51 
non-null     float64
 21  State Demographics.Miscellaneous.Under 65 Years Witout Health Insurance  51 
non-null     float64
 22  State Demographics.Miscellaneous.Under 66 Years With a Disability     51 
non-null     float64
 23  State Demographics.Sales.Accommodation and Food Services Sales        51 
non-null     float64
 24  State Demographics.Sales.Retail Sales                                 51 
non-null     float64
dtypes: float64(24), object(1)
memory usage: 10.1+ KB
actual [0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1]
```

```
predicted [0 1 0 0 1 1 1 0 1 0 1]
Accuracy 81.81818181818181
```

In [364…

```
#Prediction using Random Forest model & Feature Importance
```

In [365…

```
pipeline = Pipeline(steps=[
    ("preprocessor", preprocess),
    ("classifier", RandomForestClassifier(n_estimators=1000, random_state=40))
])

# Fit model
pipeline.fit(X_train, y_train)

# Predict
y_pred = pipeline.predict(X_test)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))

r = permutation_importance(pipeline, X_test, y_test, n_repeats=5, random_state=4
perm_df = pd.DataFrame({"feature":X_test.columns, "importance":r.importances_mea
perm_df.head(10)
```

```
Accuracy: 0.8181818181818182
```

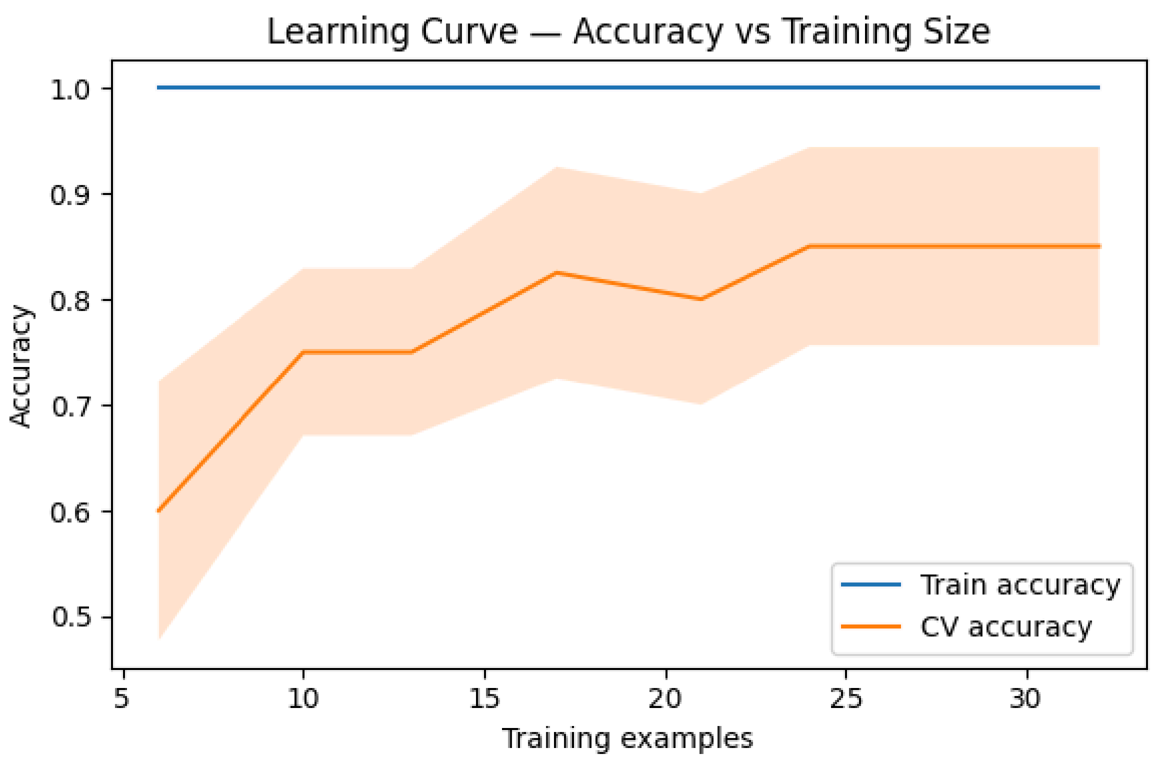
Out[365…

	feature	importance
20	State Demographics.Miscellaneous.Language Othe...	0.145455
22	State Demographics.Miscellaneous.Under 66 Year...	0.109091
3	Social Determinants.Consumed 1 or More Fruits ...	0.090909
6	Social Determinants.Healthy Weight (BMI 18.5 t...	0.072727
1	Cigarette Use.Adult.Smokers	0.072727
14	State Demographics.Ethnicities.Hispanic	0.072727
11	State Demographics.Education.Bachelor's Degree...	0.018182
19	State Demographics.Income.Persons Below Povert...	0.018182
16	State Demographics.Housing.Households with Int...	0.018182
8	Social Determinants.Received at Least One Reco...	0.000000

Learning Curve showing accuracy segregates around 81%

In [366…

```
import numpy as np
train_sizes = np.linspace(0.2, 1.0, 8)
sizes, train_scores, val_scores = learning_curve(
    estimator=pipeline,
    X=X_train,
    y=y_train,
    train_sizes=train_sizes,
    cv=5,
    scoring="accuracy",
    n_jobs=-1,
    shuffle=True,
    random_state=42,
)

train_mean = train_scores.mean(axis=1)
train_std  = train_scores.std(axis=1)
val_mean   = val_scores.mean(axis=1)
val_std    = val_scores.std(axis=1)

plt.figure(figsize=(6,4))
plt.plot(sizes, train_mean, label="Train accuracy")
plt.fill_between(sizes, train_mean-train_std, train_mean+train_std, alpha=0.2)
plt.plot(sizes, val_mean, label="CV accuracy")
plt.fill_between(sizes, val_mean-val_std, val_mean+val_std, alpha=0.2)
plt.title("Learning Curve — Accuracy vs Training Size")
plt.xlabel("Training examples")
plt.ylabel("Accuracy")
plt.legend()
plt.tight_layout()
plt.show()
```

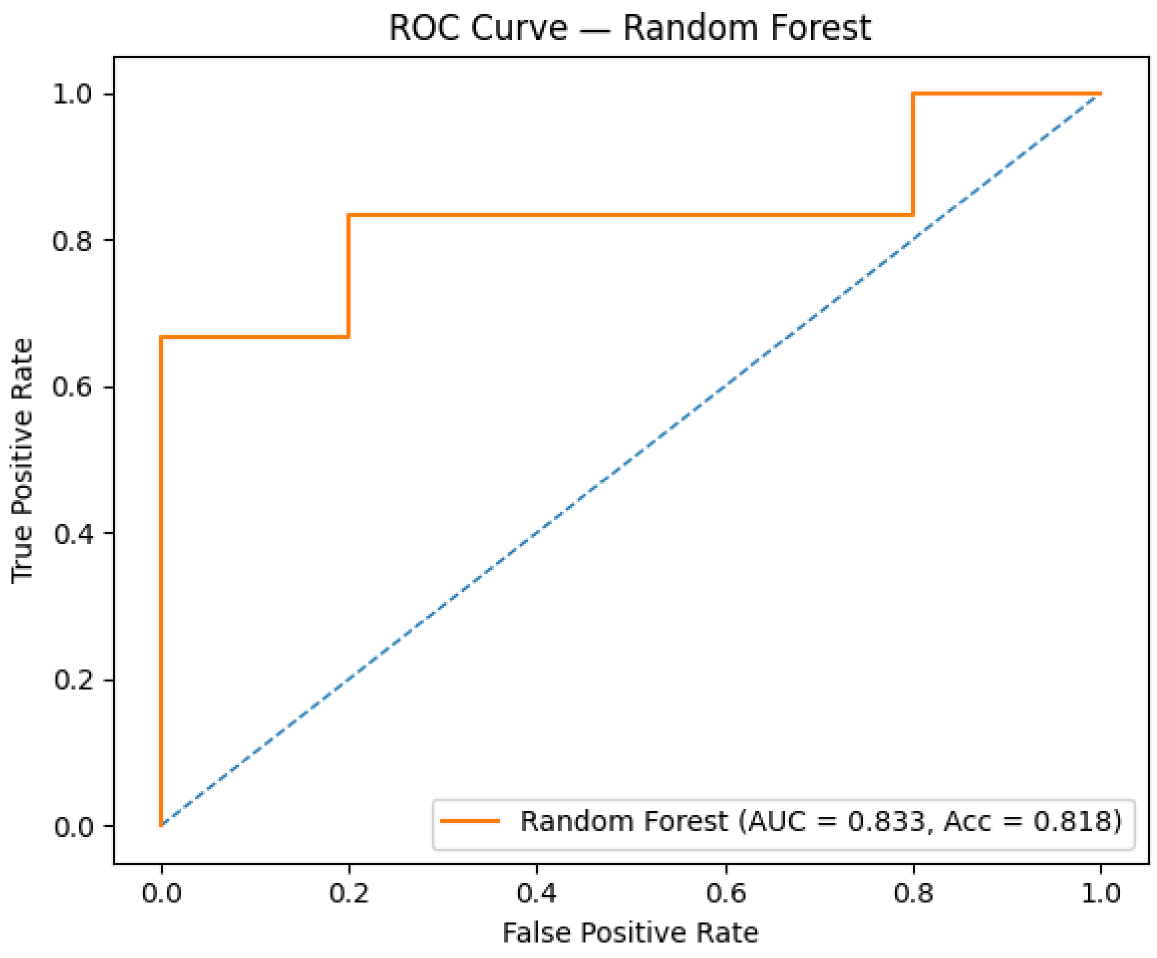


ROC AUC Curve showing 83% TP rate at 20% FP

Random Forest achieved ROC-AUC = 0.83, meaning it correctly ranks a true cancer state above a false cancer one about 83% of the time. Using the ROC curve, we select a threshold near the top-left to capture most true positives while keeping false positives low

In [367…
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score, accuracy_score
```

```python
# Probabilities for positive class on the test set
y_score = pipeline.predict_proba(X_test)[:, 1]

# ROC points and AUC
fpr, tpr, _ = roc_curve(y_test, y_score)
auc = roc_auc_score(y_test, y_score)

# (optional) accuracy at 0.5 threshold for reference
y_pred = (y_score >= 0.5).astype(int)
acc = accuracy_score(y_test, y_pred)

plt.figure(figsize=(6,5))
plt.plot([0,1], [0,1], "--", linewidth=1)  # chance
plt.plot(fpr, tpr, label=f"Random Forest (AUC = {auc:.3f}, Acc = {acc:.3f})")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve — Random Forest")
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()

# Save high-res image for PPT if you like:
# plt.savefig("roc_rf.png", dpi=300, bbox_inches="tight")
```

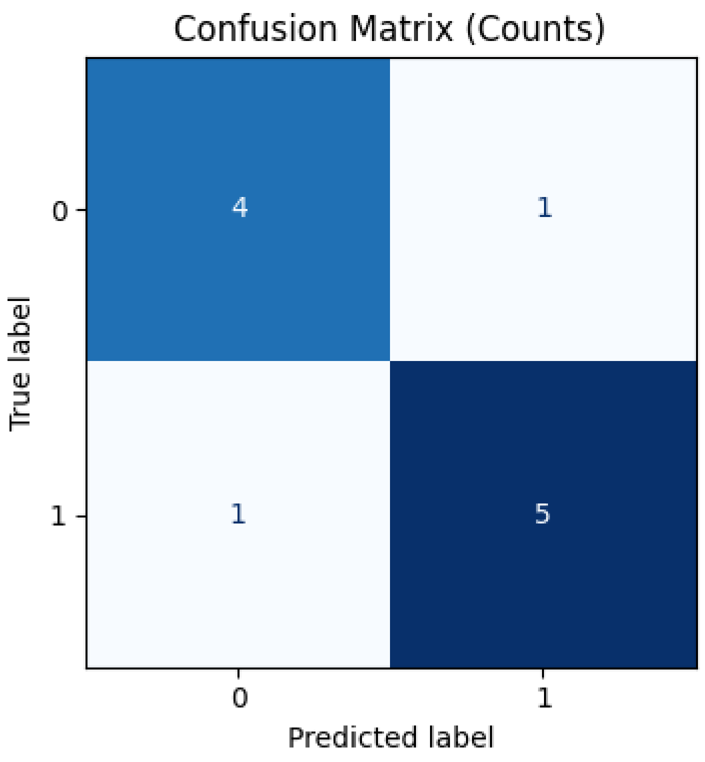


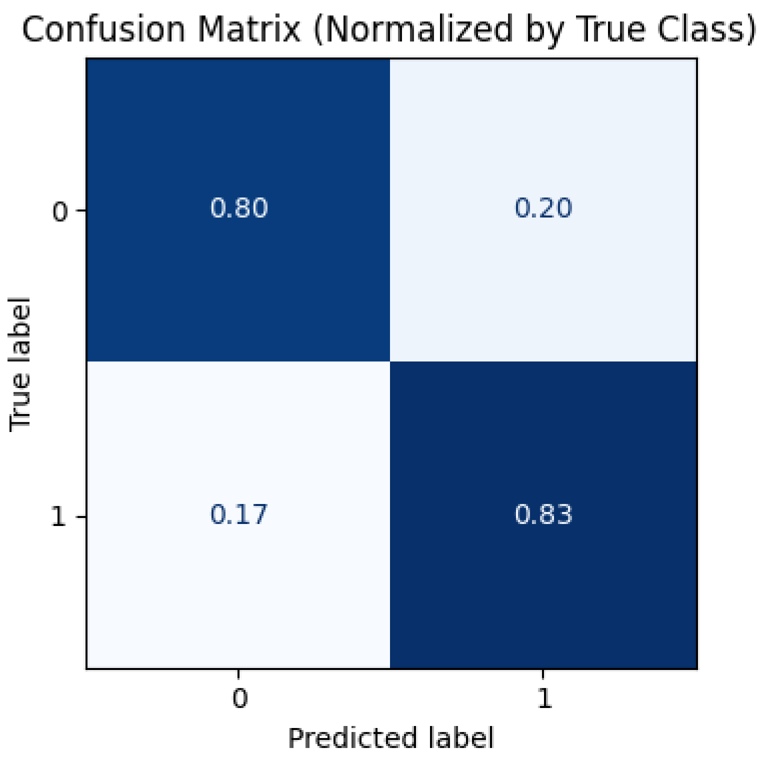
Confusion Matrix of the predicted result showing 83 percent TP cases.

In [368…

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_s
import matplotlib.pyplot as plt
import pandas as pd

# If you already have predictions:
# y_pred = pipeline.predict(X_test)   # or model.predict(X_test)

# 1) Raw counts
cm = confusion_matrix(y_test, y_pred)
tn, fp, fn, tp = cm.ravel()
print({"TN": tn, "FP": fp, "FN": fn, "TP": tp})

# 2) Nice table (counts)
cm_tbl = pd.DataFrame(cm,
    index=["Actual 0", "Actual 1"],
    columns=["Pred 0", "Pred 1"]
)
print(cm_tbl)

# 3) Matplotlib plot (counts)
fig, ax = plt.subplots(figsize=(4,4))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0,1])
disp.plot(ax=ax, cmap="Blues", values_format="d", colorbar=False)
ax.set_title("Confusion Matrix (Counts)")
plt.tight_layout()
plt.show()

# 4) Normalized version (rates by true class)
cm_norm = confusion_matrix(y_test, y_pred, normalize="true")
fig, ax = plt.subplots(figsize=(4,4))
disp = ConfusionMatrixDisplay(confusion_matrix=cm_norm, display_labels=[0,1])
disp.plot(ax=ax, cmap="Blues", values_format=".2f", colorbar=False)
ax.set_title("Confusion Matrix (Normalized by True Class)")
plt.tight_layout()
plt.show()

# (Optional) quick metrics for the slide footer
print({
    "Accuracy": round(accuracy_score(y_test, y_pred), 3),
    "Precision": round(precision_score(y_test, y_pred), 3),
    "Recall": round(recall_score(y_test, y_pred), 3),
    "F1": round(f1_score(y_test, y_pred), 3),
})
```

```
{'TN': np.int64(4), 'FP': np.int64(1), 'FN': np.int64(1), 'TP': np.int64(5)}
          Pred 0  Pred 1
Actual 0       4       1
Actual 1       1       5
```





```
{'Accuracy': 0.818, 'Precision': 0.833, 'Recall': 0.833, 'F1': 0.833}
```

We got almost the same prediction accuracy 81% with both Logistic and Random forest model showcasing liniearity in data
Furthermore we found Consuming one or more fruits, Cigratte use and BMI had major importance in determining cancer death result.

Our EDA established a clean, state-level table with 51 complete observations across 24 numeric indicators spanning social determinants (fruit/vegetable intake, smoking, BMI, screening, physical activity), demographics (age structure, education, ethnicity) metrics;

all listed columns show 51 non-null values, confirming no missingness, with numeric datatype for features, making the dataset tidy and modeling-ready.

Because death rates were reported per 100,000, we also converted numeric death-rate fields to a per-1,000 scale to keep magnitudes intuitive and comparable across analyses.

The EDA shows that Eastern US region is more prone to cancer death, vermont facing the largest number of deaths with cancer amongst all USA states.

Moving from exploration to inference, a one-way ANOVA testing whether Region is associated with Cancer Death Rate (Total) yielded F = 3.5 with p = 0.006, which is well below α = 0.05; this indicates a statistically significant regional signal in mortality burden. Given this result, we ran a post-hoc Tukey HSD to localize which regional pairs differ, operationalized on a tidy frame of death rates and region labels—appropriate because ANOVA assumptions were already met at the summary level and we sought controlled pairwise contrasts.

For prediction, we framed a supervised task by binarizing the outcome at the mean (above-mean = 1, below-mean = 0), then explicitly dropped all raw death-rate columns from the features to prevent leakage before modeling—a small but crucial detail to preserve honest generalization.

A baseline Random Forest within a preprocessing pipeline achieved ~81.8% accuracy on held-out predictions, and a logistic regression baseline reached a nearly identical score (~81%), suggesting largely linear separability with modest non-linear gains—useful for stakeholders seeking interpretable, stable rules without major performance trade-offs.

INSIGHTS:
The forest's feature-importance profile aligns with public-health priors and the EDA:
The most influential predictors included Under 66 with a disability (~0.109), Consumed 1+ fruits/day (~0.091), Healthy-weight BMI (~0.073), Adult smokers (~0.073), and Hispanic ethnicity (~0.073), with smaller weights distributed across education, poverty, and internet access—together sketching a socio-behavioral and access-to-care footprint linked to mortality risk.

Interpreting these pieces together:
(1) the EDA shows a well-structured, comprehensive panel of behavioral and demographic indicators available for every state.
(2) inferential testing confirms meaningful geographic (regional) differences in mortality rates beyond noise.
(3) the ML layer highlights actionable levers—tobacco exposure, diet quality, body-weight distribution and disability context—that can help prioritize interventions and tailor outreach.

Practically, this means public-health planners can use regional findings to target resources at high-risk areas and leverage the importance profile to choose specific, measurable program levers (e.g., smoking cessation, produce access, preventive-care navigation in diverse communities), while the parity between logistic and random-forest performance offers flexibility: deploy a simple, transparent logistic model for policy communication, or keep the random forest model for robustness without sacrificing interpretability provided by ranked importances.