# Obesity Data Analysis Project - Prediciting Obesity

**Peter Gray**

September 4, 2025

Produced with Quarto and Python

# Table of contents

## List of Python and R Packages

| Python |
|---|
| NumPy |
| Pandas |
| Matplotlib |
| seaborn |
| sklearn |
| statsmodels.api |

# Data Loading and Wrangling

Master dataframe is created in the data analysis project href here

```
import pandas as pd
import numpy as np

df =
↪  pd.read_csv(r"/home/pgr16/Documents/Data_Analysis/Obesity_Analysis_and_Prediction/Data/Master
```

### Data Wrangling and BMI Age Group Variable Creation

```
df['HEIGHT'] = df['HEIGHT']/100 # convert to meteres

df['BMI'] = df['WEIGHT'] / (df['HEIGHT']**2)
df['Obese_Y_N'] = np.where(df['BMI'] >= 30, 1, 0)
```

# Logistic Reegression

### Variable Selection

```
y_var = df["Obese_Y_N"]   # outcome
X_var = df[["AGE_GROUP", "GENDER", "RACE", "BIRTH_COUNTRY",
        "SALT_IN_PREP", "NUMBER_FOOD_DRINKS", "CALORIES",
        "NUT_CARBS_GRAMS", "NUT_FAT_GRAMS",
        "NUT_CAFFEINE_MILIGRAMS", "NUT_ALCOHOL_GRAMS",
        "TOTAL_WATER_DRANK_LITRE"]]

X_var = pd.get_dummies(X_var, drop_first=True)

# Drop the NAs
```

### Remove the NAs

```
X_var = X_var.dropna()
y_var = y_var.loc[X_var.index]
```

### Creation of Training Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_var, y_var,
↪  test_size=0.3, random_state=42)
```

## Logistic Regression Model 1

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

logmod = LogisticRegression(max_iter=10000, random_state=0)

logmod.fit(X_train, y_train)

acc = accuracy_score(y_test, logmod.predict(X_test)) *100

print(f"Logistic Regression model accuracy: {acc:.2f}%")
```

```
Logistic Regression model accuracy: 68.09%
```

### Coefficients Logistic Regression Model 1

```
coefficients = pd.DataFrame({
    'Feature': X_train.columns,
    'Coefficient': logmod.coef_[0]
})

# Optional: Sort by absolute value of coefficients
coefficients['Abs_Coefficient'] = coefficients['Coefficient'].abs()
coefficients = coefficients.sort_values(by='Abs_Coefficient', ascending=False)
coefficients
```

| | Feature | Coefficient | Abs_Coefficient |
|---|---|---|---|
| 9 | AGE_GROUP_Less than 18 | -2.000270 | 2.000270 |
| 14 | RACE_Other Race - Including Multi-Racial | -0.908169 | 0.908169 |
| 12 | RACE_Non-Hispanic White | -0.585536 | 0.585536 |
| 18 | SALT_IN_PREP_Rarely | 0.571331 | 0.571331 |
| 17 | SALT_IN_PREP_Occasionally | 0.558330 | 0.558330 |

| | Feature | Coefficient | Abs_Coefficient |
|---|---|---|---|
| 7 | AGE_GROUP_36–60 | 0.514841 | 0.514841 |
| 16 | SALT_IN_PREP_Never | 0.445018 | 0.445018 |
| 19 | SALT_IN_PREP_Very often | 0.433826 | 0.433826 |
| 15 | BIRTH_COUNTRY_US | 0.388024 | 0.388024 |
| 8 | AGE_GROUP_Above 60 | 0.382325 | 0.382325 |
| 10 | GENDER_Male | -0.213255 | 0.213255 |
| 13 | RACE_Other Hispanic | -0.209125 | 0.209125 |
| 11 | RACE_Non-Hispanic Black | -0.122874 | 0.122874 |
| 6 | TOTAL_WATER_DRANK_LITRE | 0.066987 | 0.066987 |
| 0 | NUMBER_FOOD_DRINKS | -0.054301 | 0.054301 |
| 5 | NUT_ALCOHOL_GRAMS | -0.003438 | 0.003438 |
| 3 | NUT_FAT_GRAMS | 0.002851 | 0.002851 |
| 2 | NUT_CARBS_GRAMS | -0.001171 | 0.001171 |
| 4 | NUT_CAFFEINE_MILIGRAMS | 0.000537 | 0.000537 |
| 1 | CALORIES | 0.000073 | 0.000073 |

**Confusion Matrix - Logistic Regression Model 1**

```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns


# Predictions
y_pred_train  = logmod.predict(X_train)

conf_matrix_train = confusion_matrix(y_train, y_pred_train)
conf_matrix_train

sns.heatmap(conf_matrix_train, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

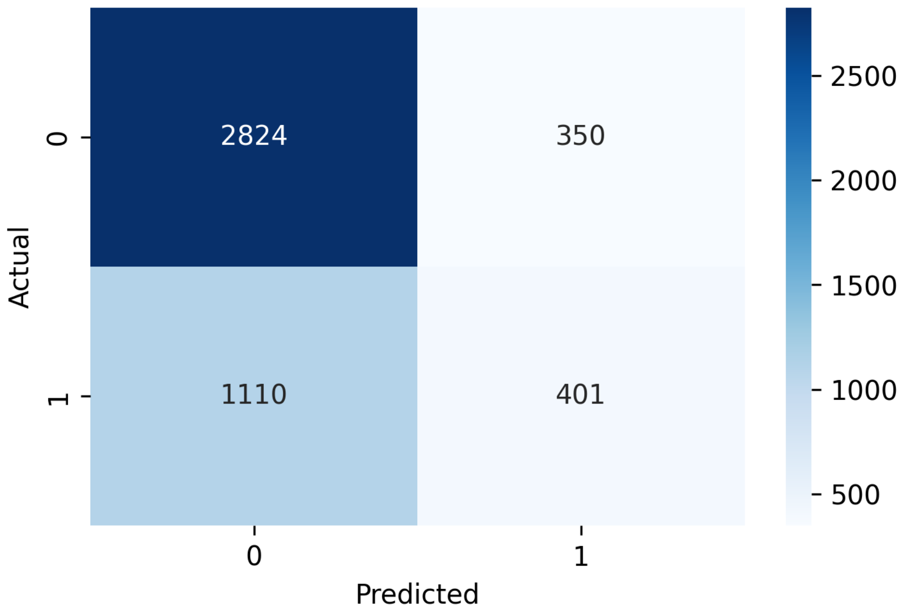



Figure 1: Confusion Matrix for Logistic Regression Model 1

**Classification Report Logistic Regression Model**

```
print(classification_report(y_train, y_pred_train))
```

```
              precision    recall  f1-score   support

           0       0.72      0.89      0.79      3174
           1       0.53      0.27      0.35      1511

    accuracy                           0.69      4685
   macro avg       0.63      0.58      0.57      4685
weighted avg       0.66      0.69      0.65      4685
```

```
from sklearn.metrics import roc_curve, roc_auc_score, RocCurveDisplay

y_proba_test = logmod.predict_proba(X_test)[:, 1]


fpr, tpr, _ = roc_curve(y_test, y_proba_test)
auc_score = roc_auc_score(y_test, y_proba_test)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {auc_score:.2f})")

plt.plot([0, 1], [0, 1], "k--", label="Random")

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

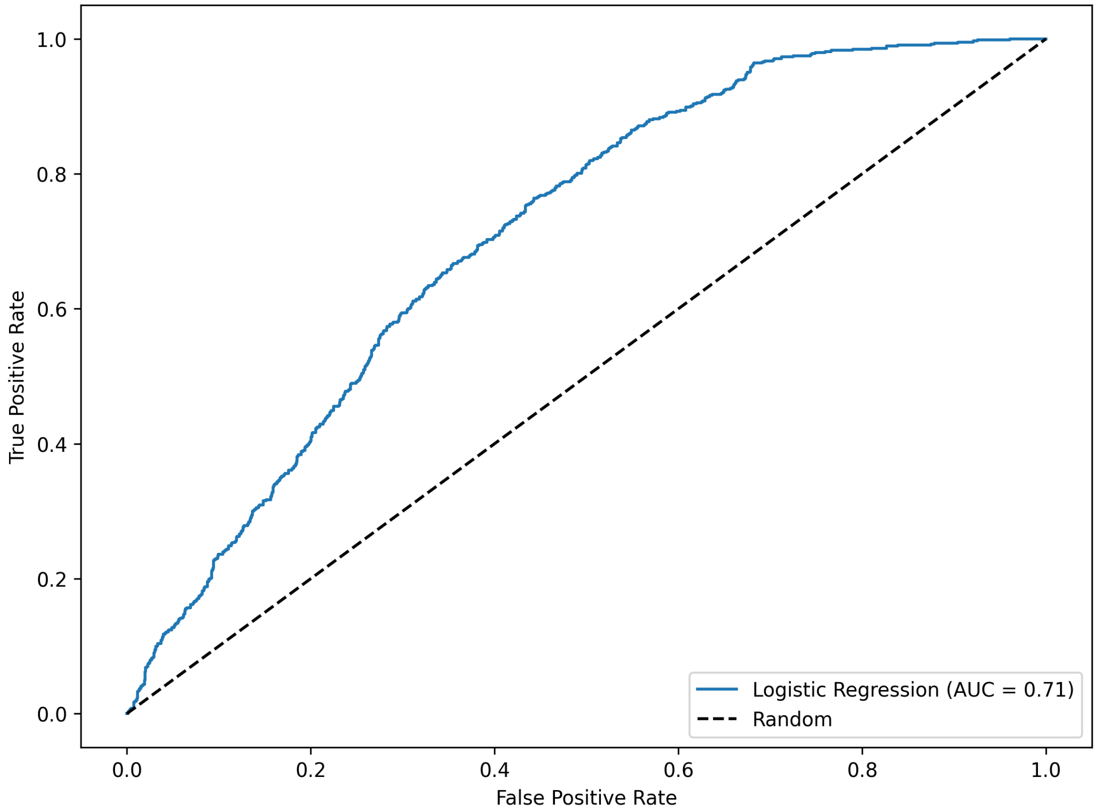



Figure 2: ROC Curve for Logistic Regression Model 1

Since the Model is OK, but could we better? We will try a LASSO regression and a Random Forest Plot to see if we can get a better model.

## LASSO Rgression

```
lasso = LogisticRegression(penalty = 'l1', solver = 'liblinear',
↪  class_weight='balanced', C=1.0)
lasso.fit(X_train, y_train)

coefs = lasso.coef_[0]  # Get coefficients
feature_names = X_train.columns


coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefs})

kept = coef_df[coef_df['Coefficient'] != 0].sort_values(by='Coefficient',
↪  key=abs, ascending=False)
discarded = coef_df[coef_df['Coefficient'] == 0]

report_dict = classification_report(y_test, lasso.predict(X_test),
↪  output_dict=True)
class_report = pd.DataFrame(report_dict).transpose()
```

```
y_proba_lasso_test = lasso.predict_proba(X_test)[:, 1]
fpr_lasso, tpr_lasso, _ = roc_curve(y_test, y_proba_lasso_test)
auc_lasso = roc_auc_score(y_test, y_proba_lasso_test)

plt.figure(figsize=(8, 6))
plt.plot(fpr_lasso, tpr_lasso, label=f"Logistic Regression (AUC =
↪  {auc_lasso:.2f})")

plt.plot([0, 1], [0, 1], 'k--', label='Random')

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

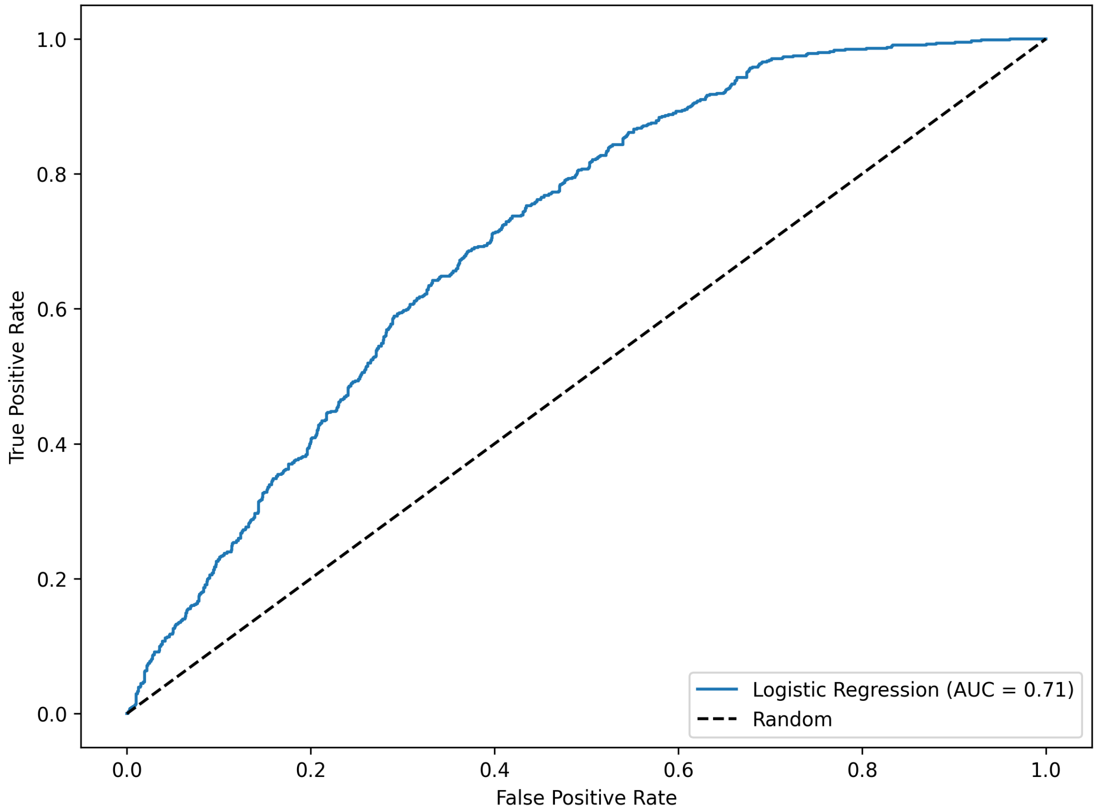



Figure 3: ROC Curve for the Logistic Regression Model 1

**Discarded Features**

```
plt.figure(figsize=(10, 6))
plt.barh(kept['Feature'], kept['Coefficient'], color='darkorange')
plt.xlabel("Coefficient Value")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

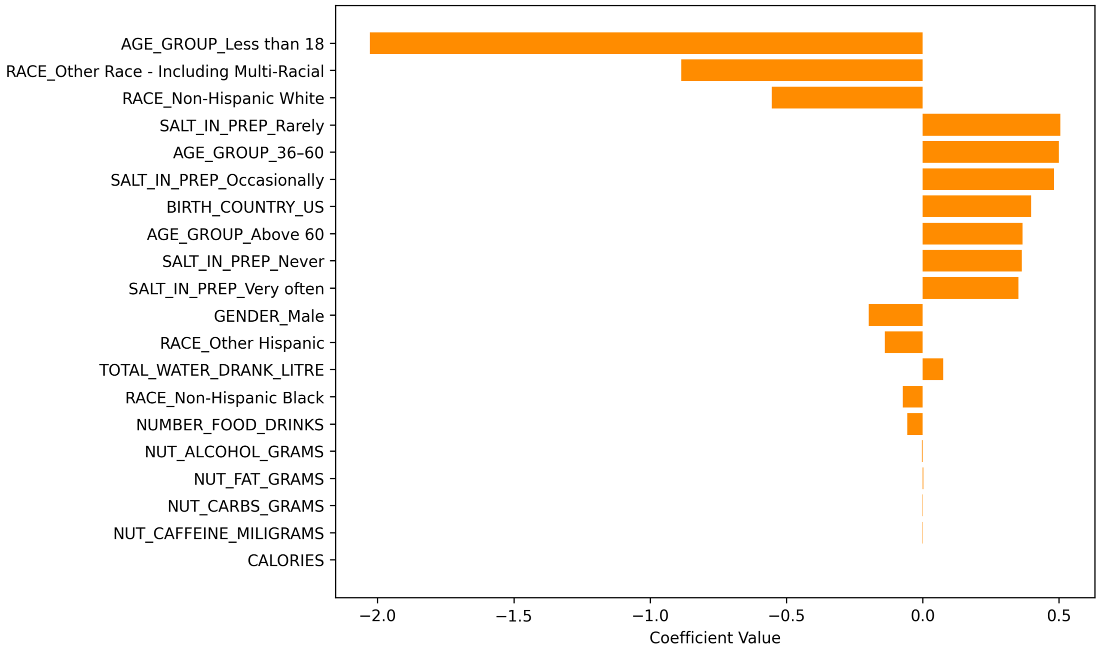



Figure 4: LASSO-Selected Feature Coefficients

**Comparison of Logistc and LASSO ROC curve**

```
plt.figure(figsize=(8, 6))
plt.plot(fpr_lasso, tpr_lasso, label=f"LASSO Logistic (AUC = {auc_lasso:.2f})")
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {auc_score:.2f})")
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

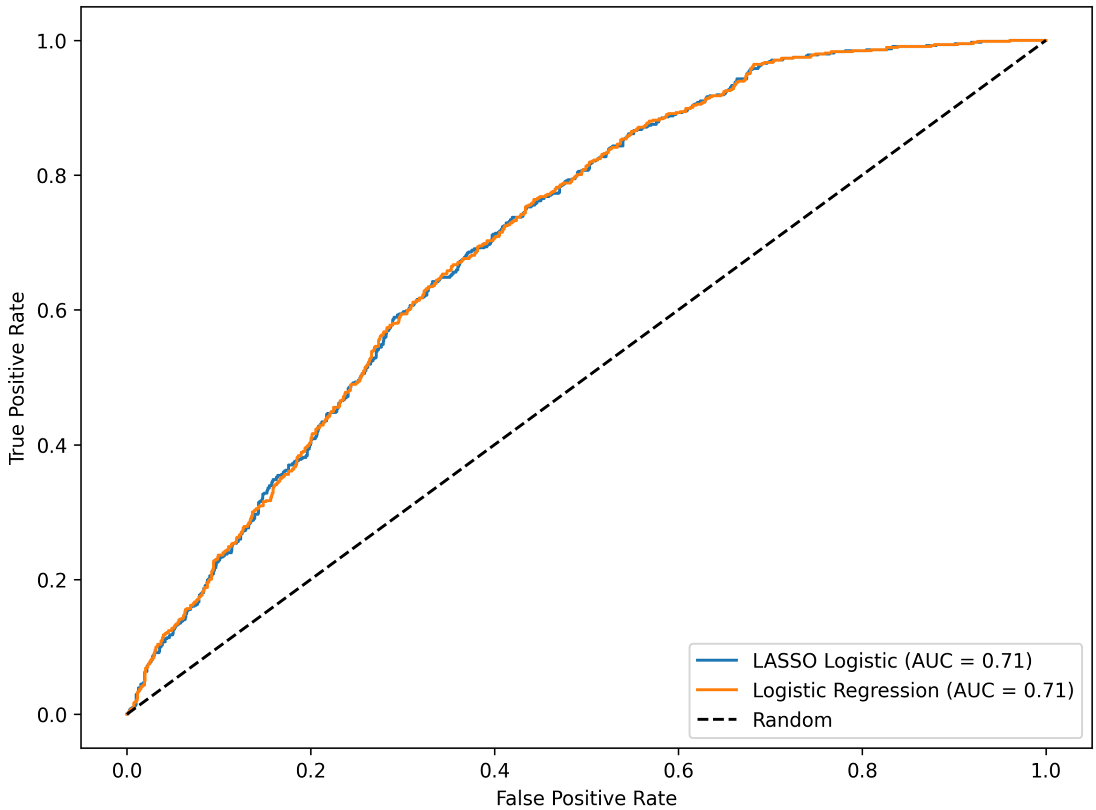



Figure 5: ROC Curve comparison between LASSO and Logistic Regression Model

## Random Forest Plot

```
from sklearn.ensemble import RandomForestClassifier as randforest

rf = randforest(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)


y_proba_randforest_test = rf.predict_proba(X_test)[:, 1]
fpr_randforest, tpr_randforest, _ = roc_curve(y_test, y_proba_randforest_test)
auc_randforest = roc_auc_score(y_test, y_proba_randforest_test)
```

```
plt.figure(figsize=(8, 6))
plt.plot(fpr_randforest, tpr_randforest, label=f"Random Forest (AUC =
 ↪ {auc_randforest:.2f})")

plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

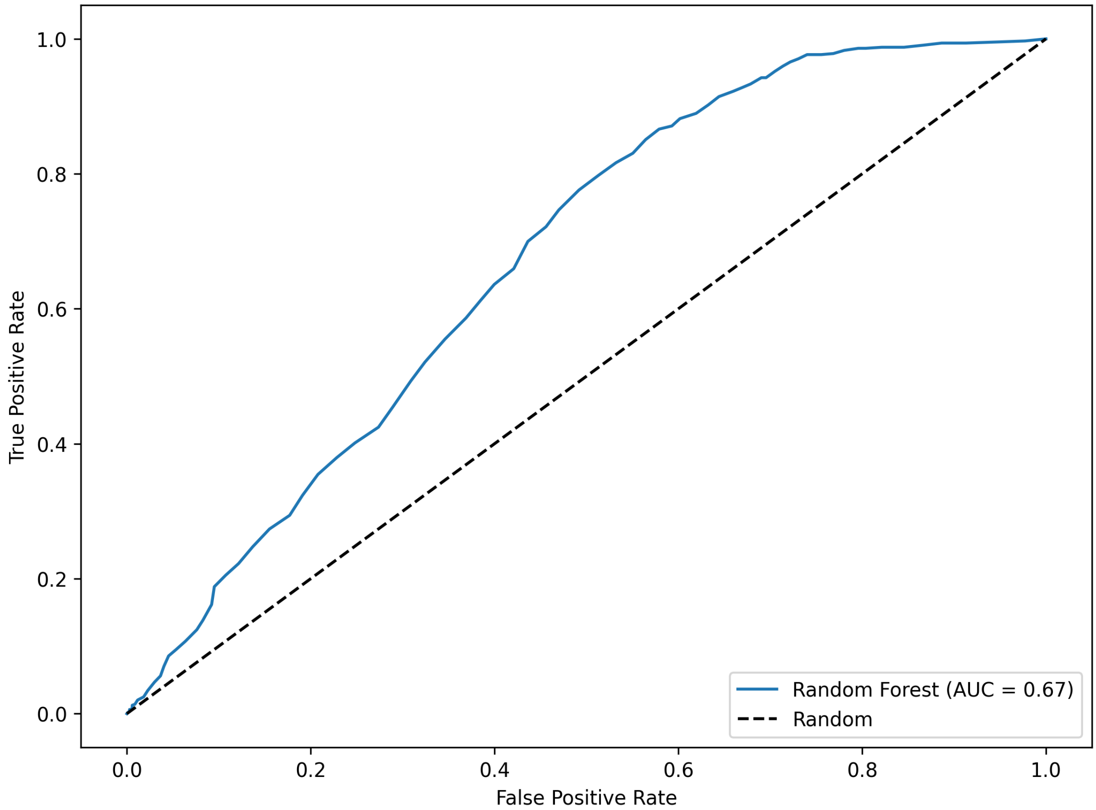



Figure 6: ROC Curve for Random Forest Model

**Comparison of all 3 models**

**Comparison of Logistc, Radnom Forst and LASSO ROC curve**

```
plt.figure(figsize=(8, 6))
plt.plot(
    fpr_randforest, tpr_randforest, label=f"Random Forest (AUC =
 ↪  {auc_randforest:.2f})"
)
plt.plot(fpr_lasso, tpr_lasso, label=f"LASSO Logistic (AUC = {auc_lasso:.2f})")
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {auc_score:.2f})")
plt.plot([0, 1], [0, 1], "k--", label="Random")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

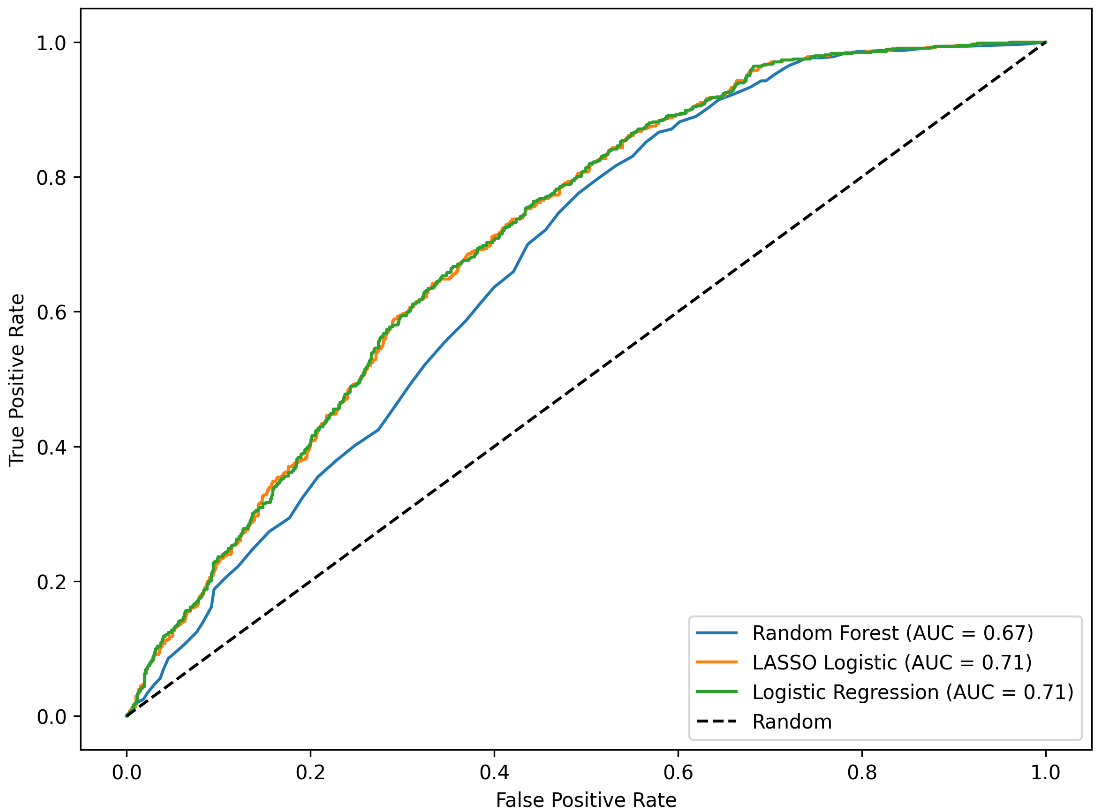



Figure 7: ROC Curve comparison between LASSO and Logistic Regression Model and Random Forest Model

# Models removing colinearity

## Colinearity

```
corr = X_var.corr()
plt.figure(figsize=(12,8))
sns.heatmap(corr, annot=False, cmap="coolwarm", center=0)
plt.show()
```

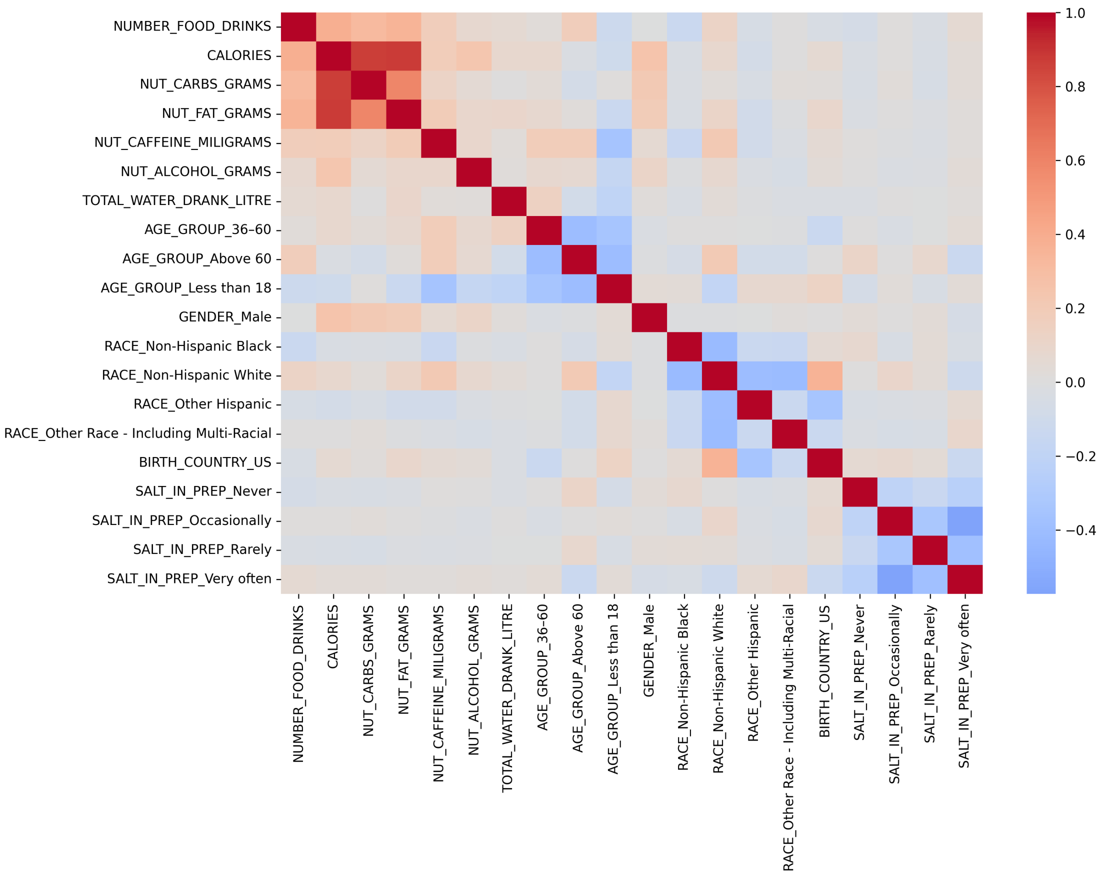



Figure 8: Correlation Matrix of Predictors

### Identifying Columns to Drop

```
upper_tri = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool))
```

```
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] >
↪  0.85)]


print(to_drop)
```

```
['NUT_CARBS_GRAMS', 'NUT_FAT_GRAMS']
```

'NUT_CARBS_GRAMS' and 'NUT_FAT_GRAMS' are identified as highly colinear ( >0.85) and therefore we will remove them to see if there is any improvement in the models we run before

```
df2 = df.drop(to_drop, axis = 1)
```

```
y_var = df2["Obese_Y_N"]   # outcome
X_var = df2[["AGE_GROUP", "GENDER", "RACE", "BIRTH_COUNTRY",
        "SALT_IN_PREP", "NUMBER_FOOD_DRINKS", "CALORIES",
        "NUT_CAFFEINE_MILIGRAMS", "NUT_ALCOHOL_GRAMS",
        "TOTAL_WATER_DRANK_LITRE"]]

X_var = pd.get_dummies(X_var, drop_first=True)

# Drop the NAs
```

## Remove the NAs

```
X_var = X_var.dropna()
y_var = y_var.loc[X_var.index]
```

### Confusion Matrix - Logistic Regression Model 2

```
from sklearn.metrics import classification_report, confusion_matrix

# Predictions
y_pred_train  = logmod.predict(X_train)

conf_matrix_train = confusion_matrix(y_train, y_pred_train)
conf_matrix_train

sns.heatmap(conf_matrix_train, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

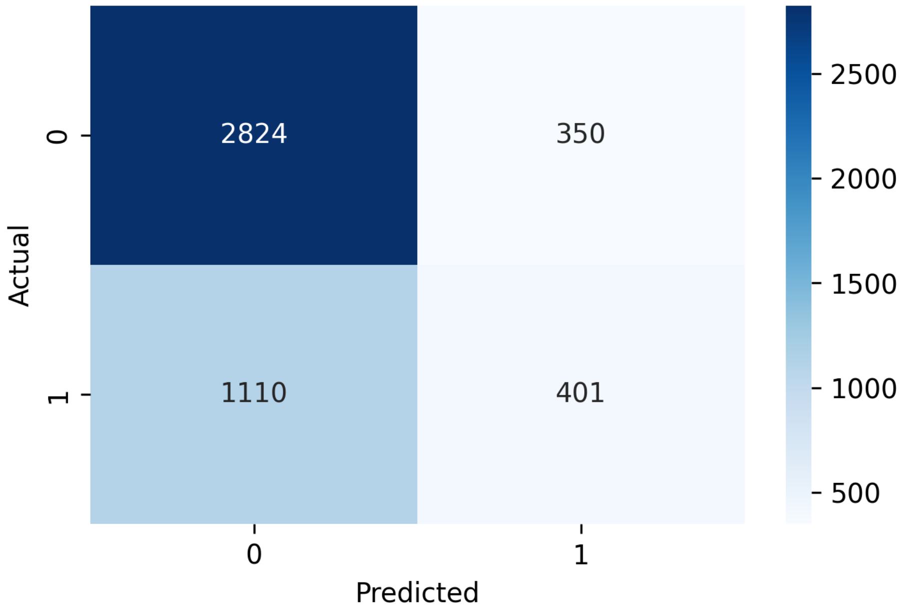



Figure 9: Confusion Matrix Logistic Regression Model 2

**Classification Report Logistic Regression Model 2**

```python
print(classification_report(y_train, y_pred_train))
```

```
              precision    recall  f1-score   support

           0       0.72      0.89      0.79      3174
           1       0.53      0.27      0.35      1511

    accuracy                           0.69      4685
   macro avg       0.63      0.58      0.57      4685
weighted avg       0.66      0.69      0.65      4685
```

```python
from sklearn.metrics import roc_curve, roc_auc_score, RocCurveDisplay

y_proba_test = logmod.predict_proba(X_test)[:, 1]


fpr, tpr, _ = roc_curve(y_test, y_proba_test)
auc_score = roc_auc_score(y_test, y_proba_test)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {auc_score:.2f})")

plt.plot([0, 1], [0, 1], 'k--', label='Random')

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

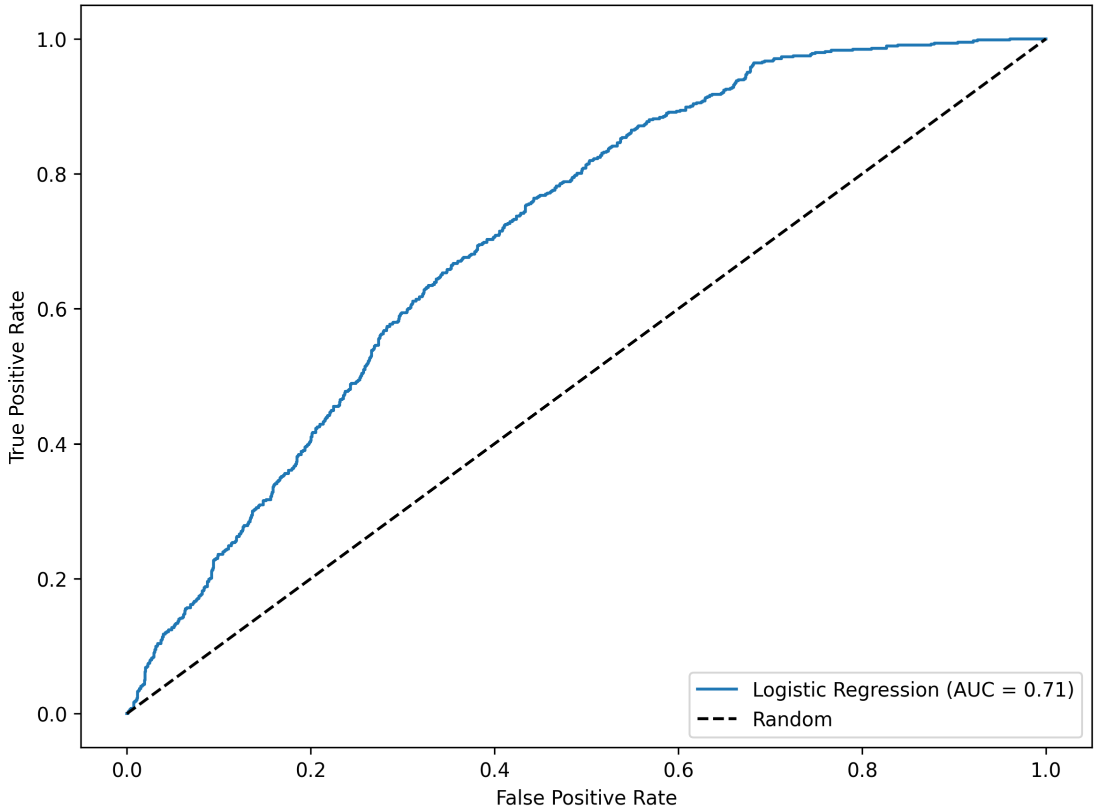



Figure 10: ROC Curve Logistic Regression Model 2

Since the Model is OK, but could we better? We will try a LASSO regression and a Random Forest Plot to see if we can get a better model.

### LASSO Rgression

```
lasso = LogisticRegression(penalty = 'l1', solver = 'liblinear',
↪  class_weight='balanced', C=1.0)
lasso.fit(X_train, y_train)

coefs = lasso.coef_[0]  # Get coefficients
feature_names = X_train.columns


coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefs})

kept = coef_df[coef_df['Coefficient'] != 0].sort_values(by='Coefficient',
↪  key=abs, ascending=False)
discarded = coef_df[coef_df['Coefficient'] == 0]

report_dict = classification_report(y_test, lasso.predict(X_test),
↪  output_dict=True)
class_report = pd.DataFrame(report_dict).transpose()
```

```
y_proba_lasso_test = lasso.predict_proba(X_test)[:, 1]
fpr_lasso, tpr_lasso, _ = roc_curve(y_test, y_proba_lasso_test)
auc_lasso = roc_auc_score(y_test, y_proba_lasso_test)

plt.figure(figsize=(8, 6))
plt.plot(fpr_lasso, tpr_lasso, label=f"Logistic Regression (AUC =
↪  {auc_lasso:.2f})")

plt.plot([0, 1], [0, 1], 'k--', label='Random')

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

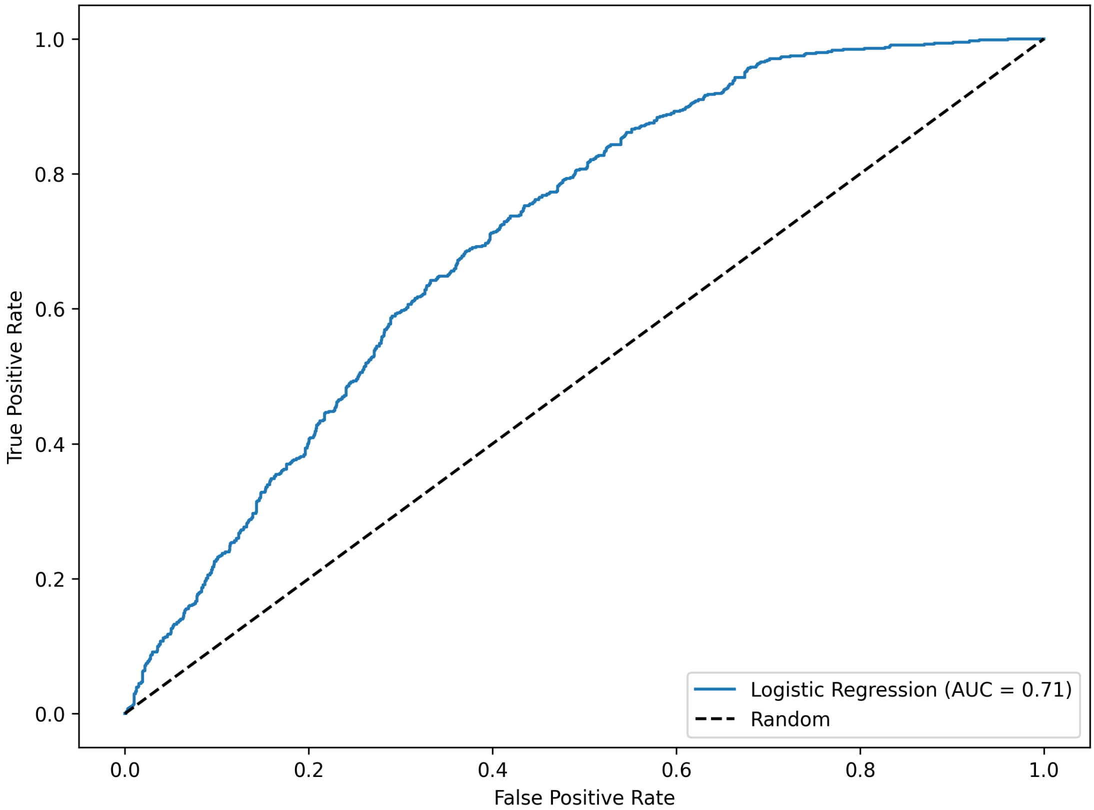



Figure 11: ROC Curve LASSO Regression Model 2

### Discarded Features

```
plt.figure(figsize=(10, 6))
plt.barh(kept['Feature'], kept['Coefficient'], color='darkorange')
plt.xlabel("Coefficient Value")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

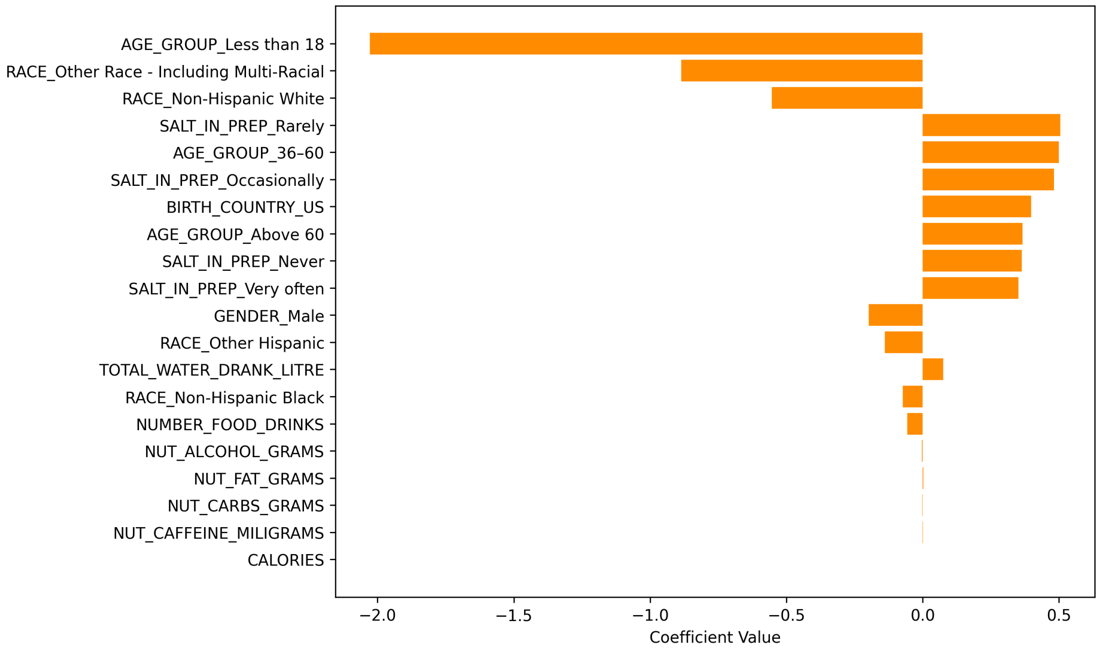



Figure 12: Discarded Features Regression Model 2

**Comparison of Logistc and LASSO ROC curve**

```
plt.figure(figsize=(8, 6))
plt.plot(fpr_lasso, tpr_lasso, label=f"LASSO Logistic (AUC = {auc_lasso:.2f})")
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {auc_score:.2f})")
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

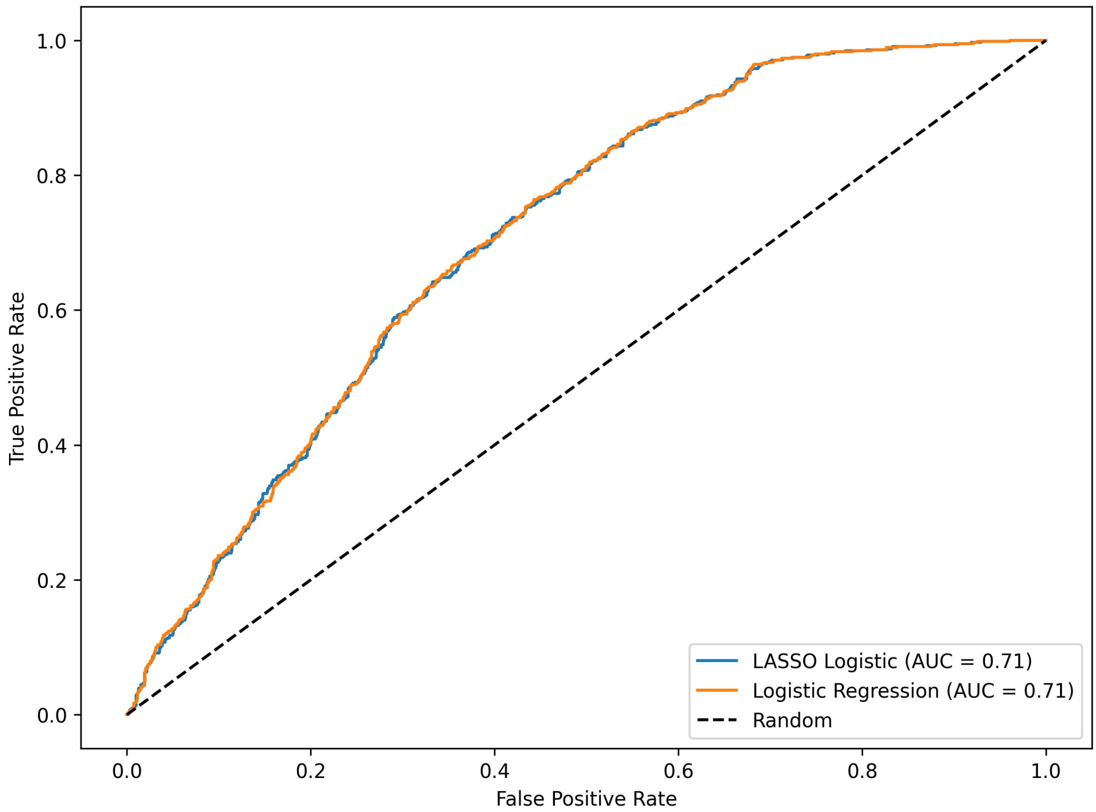



Figure 13: ROC Curve LASSO Logistic Regression

## Random Forest Plot

```
from sklearn.ensemble import RandomForestClassifier as randforest

rf = randforest(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)


y_proba_randforest_test = rf.predict_proba(X_test)[:, 1]
fpr_randforest, tpr_randforest, _ = roc_curve(y_test, y_proba_randforest_test)
auc_randforest = roc_auc_score(y_test, y_proba_randforest_test)
```

```
plt.figure(figsize=(8, 6))
plt.plot(fpr_randforest, tpr_randforest, label=f"Random Forest (AUC =
 ↪  {auc_randforest:.2f})")

plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

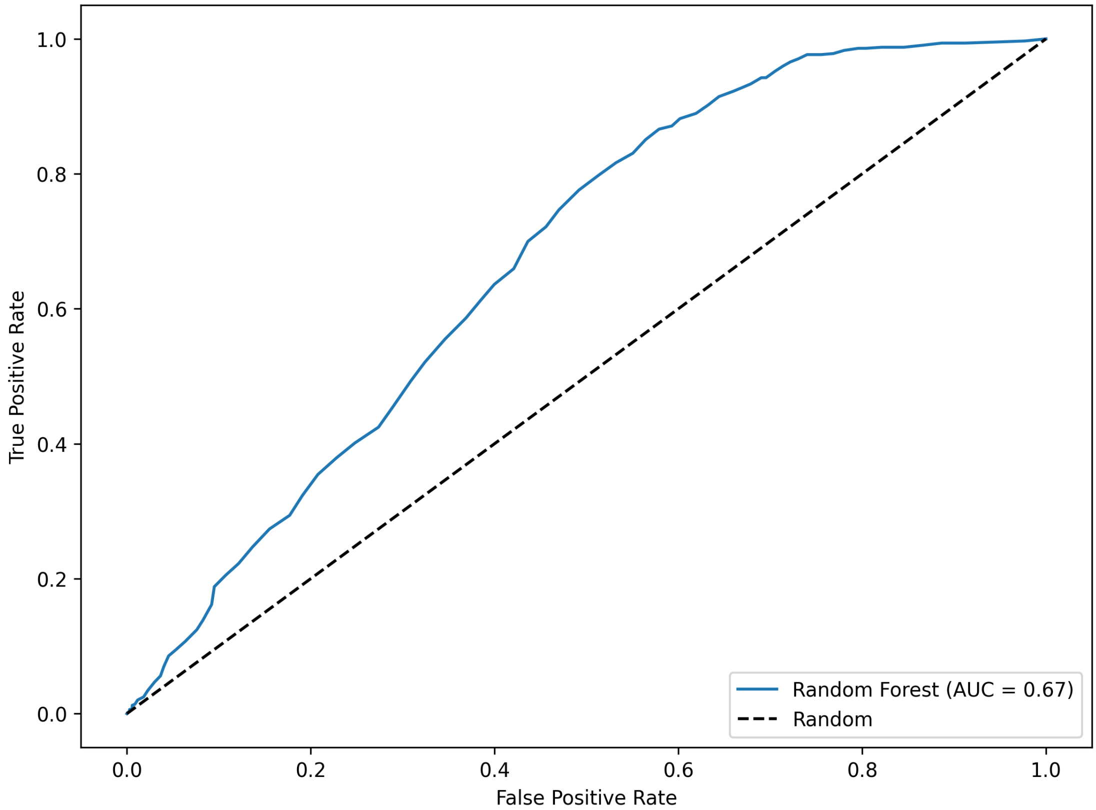



Figure 14: ROC Curve Random Forest Graph

**Comparison of all 3 models**

**Comparison of Logistc and LASSO ROC curve**

```
plt.figure(figsize=(8, 6))
plt.plot(fpr_randforest, tpr_randforest, label=f"Random Forest (AUC =
 ↪  {auc_randforest:.2f})")
plt.plot(fpr_lasso, tpr_lasso, label=f"LASSO Logistic (AUC = {auc_lasso:.2f})")
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {auc_score:.2f})")
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.grid(False)
plt.tight_layout()
plt.show()
```

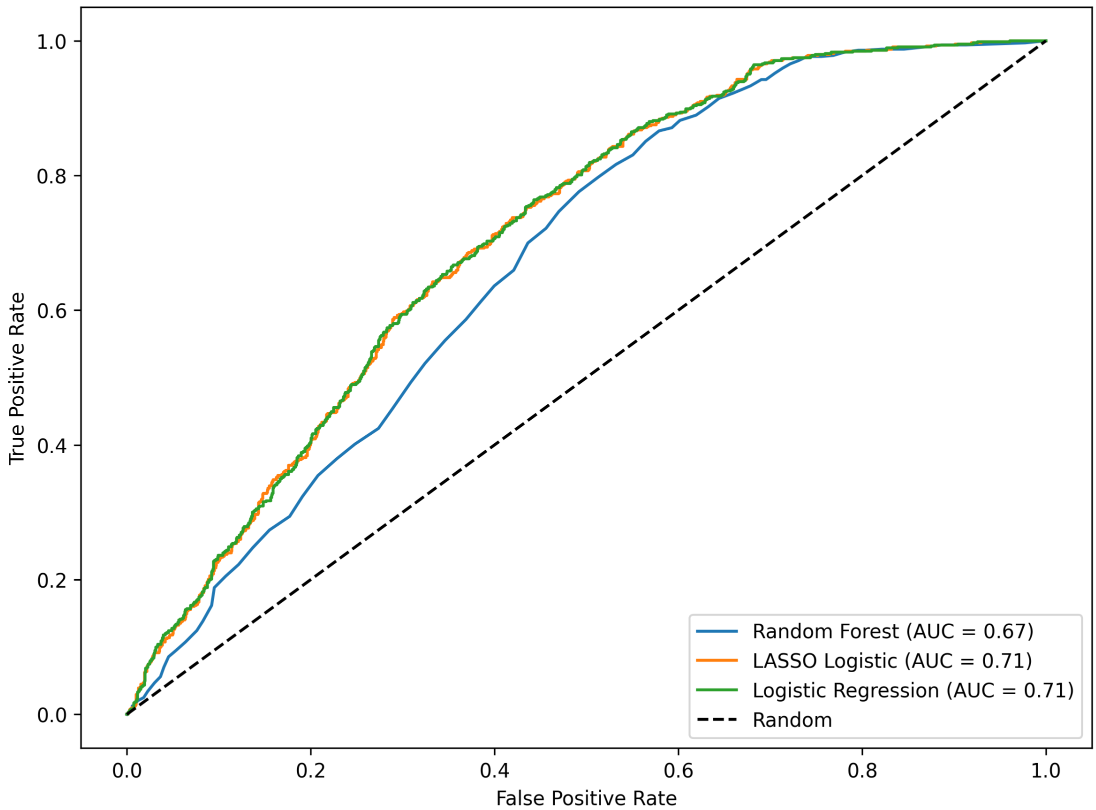



Figure 15: ROC Curve, LASSO, Random Forest Logistic Regression

There is no no real difference in the models that have highly colinear removed. We will revert back to the orignal Lgistic Regression Model with all identified .

## Conclusion from Logistic Model 1

#### Run the model agin to make sure we are not confused

We will need to provide confidence intervals. We'll do this by running the statsmodel.api rather than sci-kit-learn

```
import statsmodels.api as sm
y_var = df["Obese_Y_N"]   # outcome
X_var = df[["AGE_GROUP", "GENDER", "RACE", "BIRTH_COUNTRY",
        "SALT_IN_PREP", "NUMBER_FOOD_DRINKS", "CALORIES",
        "NUT_CARBS_GRAMS", "NUT_FAT_GRAMS",
        "NUT_CAFFEINE_MILIGRAMS", "NUT_ALCOHOL_GRAMS",
        "TOTAL_WATER_DRANK_LITRE"]]

X_var = pd.get_dummies(X_var, drop_first=True)
X_var = X_var.dropna()
y_var = y_var.loc[X_var.index]



X_sm = sm.add_constant(X_var).astype(float)
y_sm = y_var.astype(float)

# Fit logistic regression
logit_model = sm.Logit(y_var, X_sm)
result = logit_model.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.548471
         Iterations 7
```

#### Odds Ratios

```
params = result.params
conf = result.conf_int()
conf.columns = ["2.5%", "97.5%"]



or_table = pd.DataFrame({
    "Feature": params.index,
    "Coefficient": params,
    "Odds Ratio": np.exp(params),
    "2.5%": np.exp(conf["2.5%"]),
    "97.5%": np.exp(conf["97.5%"]),
```

```
    "p-value": result.pvalues})


or_table['p-value'] = np.where(or_table['p-value'] < 0.05, "*p <0.05",
↪   or_table['p-value'])
or_table = or_table[or_table["Feature"] != "const"]

or_table["Feature"] = or_table["Feature"].str.replace("_", " ",
↪   regex=False).str.title()

or_table.sort_values(by="Odds Ratio", ascending = False)
```

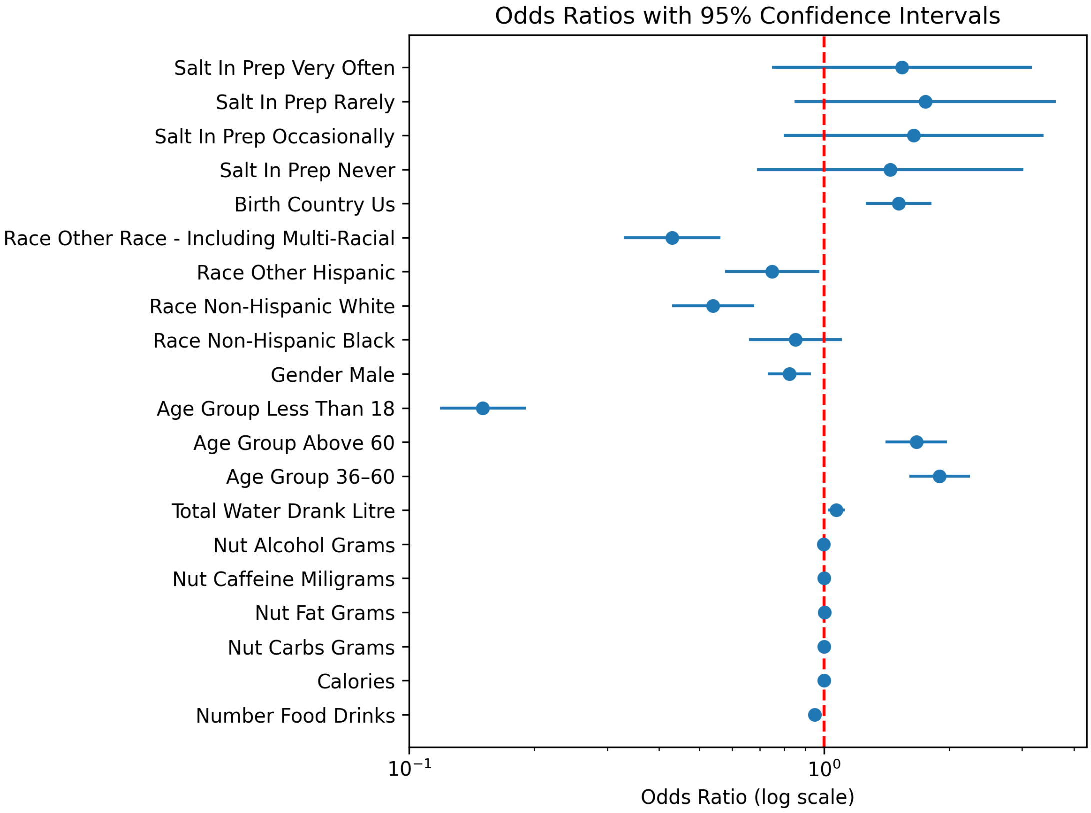
| | Feature | Coefficient | Odds Ratio |
|---|---|---|---|
| AGE_GROUP_36–60 | Age Group 36–60 | 0.639804 | 1.896110 |
| SALT_IN_PREP_Rarely | Salt In Prep Rarely | 0.560211 | 1.751042 |
| AGE_GROUP_Above 60 | Age Group Above 60 | 0.510549 | 1.666206 |
| SALT_IN_PREP_Occasionally | Salt In Prep Occasionally | 0.495983 | 1.642112 |
| SALT_IN_PREP_Very often | Salt In Prep Very Often | 0.430922 | 1.538675 |
| BIRTH_COUNTRY_US | Birth Country Us | 0.412340 | 1.510348 |
| SALT_IN_PREP_Never | Salt In Prep Never | 0.365931 | 1.441856 |
| TOTAL_WATER_DRANK_LITRE | Total Water Drank Litre | 0.067259 | 1.069573 |
| NUT_FAT_GRAMS | Nut Fat Grams | 0.002877 | 1.002881 |
| NUT_CAFFEINE_MILIGRAMS | Nut Caffeine Miligrams | 0.000389 | 1.000389 |
| CALORIES | Calories | 0.000059 | 1.000059 |
| NUT_CARBS_GRAMS | Nut Carbs Grams | -0.000989 | 0.999012 |
| NUT_ALCOHOL_GRAMS | Nut Alcohol Grams | -0.002111 | 0.997891 |
| NUMBER_FOOD_DRINKS | Number Food Drinks | -0.051876 | 0.949446 |
| RACE_Non-Hispanic Black | Race Non-Hispanic Black | -0.159616 | 0.852471 |
| GENDER_Male | Gender Male | -0.194653 | 0.823120 |
| RACE_Other Hispanic | Race Other Hispanic | -0.288626 | 0.749292 |
| RACE_Non-Hispanic White | Race Non-Hispanic White | -0.616849 | 0.539642 |
| RACE_Other Race - Including Multi-Racial | Race Other Race - Including Multi-Racial | -0.844126 | 0.429933 |
| AGE_GROUP_Less than 18 | Age Group Less Than 18 | -1.894983 | 0.150321 |

Accoridng to our model Calories alone do not necessarily appear to predict whether a person is going to be obese or not.

### Forest Plot

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.errorbar(or_table["Odds Ratio"], or_table["Feature"],
             xerr=[or_table["Odds Ratio"] - or_table["2.5%"],
                   or_table["97.5%"] - or_table["Odds Ratio"]],
             fmt='o')
plt.axvline(x=1, color="red", linestyle="--")
plt.xscale("log")
plt.xlabel("Odds Ratio (log scale)")
plt.title("Odds Ratios with 95% Confidence Intervals")
plt.tight_layout()
plt.figtext(
    0.5, -0.05,
    "Odds Ratio > 1 → Higher likelihood of Obesity | Odds Ratio < 1 → Lower
     ↪  likelihood of Obesity",
    ha="center", fontsize=10, style="italic"
)
plt.show()
```




*Odds Ratio > 1 → Higher likelihood of Obesity | Odds Ratio < 1 → Lower likelihood of Obesity*

Figure 16: FOrest Plot of Odds Ratios for Logistic Regression Model 1