PIMA INDIANS DIABETES DATASET

PYTHON ANALYSIS

EXECUTIVE SUMMARY

This project explores the relationship between biological and demographic risk factors and the likelihood of developing diabetes using logistic regression modelling on the Pima Indians Diabetes dataset. Through exploratory analysis, group comparisons, and predictive modeling, the influence of variables such as BMI, glucose, age, and pregnancy history are

explored in relation to diabetes risk. Key concepts explored include prevalence analysis, risk ratios, odds ratios, model evaluation, confounding, and ROC/AUC assessment.

The final model identifies BMI, glucose, and pregnancy history as independent predictors of diabetes. While the model has high specificity, its limited sensitivity makes it more suitable for clinical assessment than for widespread screening.

DATASET SOURCE
Dataset: Pima Indians Diabetes Dataset
Original Source: National Institute of Diabetes and Digestive and Kidney Diseases
Population: Female patients aged 21+ of Pima Indian heritage
Purpose: To diagnostically predict the presence of diabetes based on 8 medical predictors

---

SUMMARY & INTERPRETATION

This project uses the Pima Indians Diabetes Dataset to explore how biological and demographic factors influence the likelihood of having diabetes. Logistic regression and risk analysis were used to evaluate predictors such as BMI, glucose level, and pregnancy history. The goal was to identify key drivers of diabetes risk and assess the predictive power of a simple multivariable model.

The overall prevalence of diabetes in the dataset was 34.9%, more than three times higher than the general U.S. population (11.4%)[1]. Prevalence was highest among women aged 40–49 (55.1%) and 50–59 (59.6%), those who were obese (46.6%) or severely obese (46.3%), had 4 or more pregnancies (45.1%), and those with high insulin levels (54.1%).

Both crude and adjusted models showed that BMI, glucose, and pregnancies were statistically significant and independent predictors of diabetes, with odds ratios of 1.08, 1.03, and 1.15, respectively (all p-values < 0.005). These results suggest that each of these variables contributes uniquely to diabetes risk, even when accounting for the others.

In terms of model performance, the logistic regression model demonstrated strong specificity (correctly identifying non-diabetics) but moderate sensitivity (missed many true diabetic cases). The model's AUC of 0.83 indicates good overall diagnostic ability. Lowering the classification threshold from 0.50 to 0.40 improved sensitivity, meaning more true cases were detected, but this came at the cost of reduced specificity and an increase in false positives—raising ethical and practical concerns for widespread screening.

A confounding analysis between BMI, age, pregnancies, and glucose showed minimal impact. Only in the BMI + glucose model was there a small decrease in BMI's odds ratio, but not enough to suggest it was confounded. All variables remained statistically significant in two-variable models.

The final multivariable model—including BMI, glucose, and pregnancies—demonstrates that these are meaningful, independent predictors of diabetes in this population. These findings reinforce that diabetes risk is multifactorial, and clinical assessment must account for a range of interrelated health indicators.

This model may assist clinicians in identifying higher-risk patients using basic clinical and epidemiological data. However, its limited sensitivity makes it less suitable for general population screening. Future work should incorporate additional variables—such as family history, HbA1c, or physical activity levels—to enhance predictive power. Public health strategies should continue to balance the importance of early detection with the potential harms of over-diagnosis and false positives.

1. CDC. *National Diabetes Statistics Report* (2024)

MODELING PROCESS

STEP 1. IMPORT DATA & INITIAL EXPLORATION

Quick Refresh

- `.head()` shows the first 5 rows of the dataset
- `.info()` displays data types and counts
- `.describe()` shows summary statistics

CODE

```
from google.colab import files
uploaded = files.upload()

import pandas as pd

# Load the dataset
df = pd.read_csv('diabetes.csv')

# First 5 rows
```

```
df.head()

# Data types and missing values
df.info()

# Summary stats
df.describe()
```

OUTPUT

Pregnancies	Glucose	BP	Skin Thickness	Insulin	BMI	DPF	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

Variables: Pregnancies; Glucose (plasma glucose concentration); Blood Pressure (BP; mm Hg); Skin Thickness (mm); BMI (weight in kg/(height in m)^2); Diabetes Pedigree Function (DPF; a quantitative measure used to assess an individual's risk of developing type 2 diabetes based on their family history); Age (years); Outcome (diabetes, 0 = non-case, 1 = case)

REFLECTION #1

What are the variables in this dataset?

Medical measurements include glucose, blood pressure, skin thickness, insulin, and BMI. Personal/demographic variables include pregnancies, DPF, and age. The presence of diabetes is the outcome (binary).

What should be watched?

Some columns - like insulin and skin thickness - have many zeros, indicating possible missing data. The outcome variable being binary should also be noted.

Wrap-Up

The dataset contains 768 women aged 21+ from the Pima Population. There are 9 variables, all numerical, that were recorded. Further exploration of potential missing values coded as 0 should be done.

STEP 2. INITIAL CLEANING & EXPLORATION

In this step, the structure and summary statistics of the dataset are explored. This helps understanding of the range, distribution, and possible data quality issues before any modeling.

STEP 2A. EXPLORATORY DATA ANALYSIS

CODE

```
# Check structure
df.info()

# Summary statistics
df.describe()
```

OUTPUT

Variable	Count	Mean	Std Dev	Min	25%	50%	75%	Max
Pregnancies	768	3.85	3.37	0	1	3	6	17
Glucose	768	120.89	31.97	0	99	117	140.25	199
BP	768	69.11	19.36	0	62	72	80	122
Skin Thickness	768	20.54	15.95	0	0	23	32	99
Insulin	768	79.80	115.24	0	0	30.5	127.25	846
BMI	768	31.99	7.88	0	27.3	32	36.6	67.1
DPF	768	0.47	0.33	0.08	0.24	0.37	0.63	2.42
Age	768	33.24	11.76	21	24	29	41	81
Outcome	768	0.35	0.48	0	0	0	1	1

REFLECTION #2A

Do any columns stand out as possibly needing cleaning? Any outliers?

Yes - columns like glucose, BP, skin thickness, and insulin have minimum values of 0, which could indicate missing or unrecorded values.

The maximum value of insulin (846) and DPF (2.42) are higher relative to the median and 75th percentile, which could suggest possible outliers or skewed distributions.

Which variables are likely continuous vs categorical?

Continuous variables include glucose, BMI, BP, age, and insulin. Only outcome is categorical (0 or 1). Pregnancies is a discrete variable.

Wrap-Up

Several variables may contain placeholder zeros to fill in missing values, and the distributions appear non-normal and may be skewed.

STEP 2B. MISSING OR IMPLAUSIBLE VALUES

CODE

```
# Check for zero values in each column
(df == 0).sum()
```

OUTPUT

Pregnancies	111
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DiabetesPedigreeFunction	0
Age	0
Outcome	500

REFLECTION #2B

Do any columns have a lot of zeros?

Insulin has the highest number of zeros (374), while Skin Thickness has 227, and Pregnancies has 111, though zero can be valid for that variable.

Do you think those zeros represent real values or missing data?

At least for pregnancies, “0” could represent a real value. Skin Thickness, BP and insulin “0” values are not biologically possible, and so probably indicate missing data.

What should we do about them?

For now, “0” will be flagged as possibly missing, rather than dropping or replacing them. This allows analysis of their influence without introducing possible bias through imputation.

Wrap-Up

Three variables - insulin, skin thickness, and blood pressure - have been identified as likely missing values that are coded as “0”. Rather than replace or remove them, they will be flagged to explore their impact further on in the analysis.

STEP 2C. CREATING MISSINGNESS FLAGS

Quick Refresh

- Missingness flags are binary indicators that identify rows with suspicious or missing values

CODE

```
# create missingness flag
# create a new column to flag missing insulin values
df['insulin_missing'] = df['Insulin'] == 0

# preview
df[['Insulin', 'insulin_missing']].head(10)

# create new column for skin thickness
df['skin_missing'] = df["SkinThickness"] == 0

# preview
df[['SkinThickness', 'skin_missing']].head(10)
```

OUTPUT

Insulin	insulin_missing	Skin Thickness	skin_missing	BP	bp_missing
0	True	35	False	72	False
0	True	29	False	66	False
0	True	0	True	64	False
94	False	23	False	66	False

168	False	35	False	40	False

REFLECTION #2C

How might flagging missingness help?

By flagging missingness it can allow exploration of if the missingness is patterned. It may also be used as a predictor variable in a model if appropriate, along with allowing sensitivity analysis.

What's one reason you might want to impute missing values rather than just dropping them?

If imputation is done too early or without understanding the data structure, bias could be introduced. By keeping the original values, the dataset integrity and flexibility is kept.

Wrap-Up

Three flags were created for insulin, skin, and bp for missingness, which can be used later in visualizations, modeling, or sensitivity analysis.

STEP 3. PREVALENCE & GROUP COMPARISONS

In this step, how the prevalence of diabetes varies across different subgroups are explored. By binning continuous variables and calculating prevalence within each group, trends and high-risk populations can be identified.

STEP 3A. CREATING MISSINGNESS FLAGS

Quick Refresh

- Prevalence is the proportion of individuals in a population who have a condition at a specific point in time
    - Equation: (Number of people with diabetes) / (Total number of people)

CODE

```
# Prevalence
prevalence = df['Outcome'].mean() * 100
print(f"Prevalence of diabetes: {prevalence:.2f}%")
```

OUTPUT

```
Prevalence of diabetes: 34.90%
```

REFLECTION #3A

What does this tell us?

Roughly 1 in 3 women in this dataset is diabetic, which is a very high prevalence compared to general population levels. The prevalence may be higher due to oversampling individuals at risk of diabetes based on symptoms, family history, or screening criteria, etc.. It

may also be higher due to the population - Pima Indian women aged 21+ - which is known for having a higher risk of Type II Diabetes.

Wrap-Up

In the dataset, 34.9% of the participants have diabetes - confirming that diabetes is common in this cohort and reinforcing the importance of identifying risk factors.

STEP 3B. GROUP COMPARISONS

CODE

```
# Define binning
# Age
age_bins = [20, 29, 39, 49, 59, 100]
age_labels = ['20-29', '30-39', '40-49', '50-59', '60+']
df['age_group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

# BMI
bmi_bins = [0, 18.4, 24.9, 29.9, 33, 100]
bmi_labels = ['Underweight', 'Normal', 'Overweight', 'Obese', 'Severely
Obese']
df['bmi_group'] = pd.cut(df['BMI'], bins=bmi_bins, labels=bmi_labels)

# Pregnancies
preg_bins = [0, 0.5, 1.5, 2.5, 3.5, 20]
preg_labels = ['0', '1', '2', '3', '4+']
df['preg_group'] = pd.cut(df['Pregnancies'], bins=preg_bins,
labels=preg_labels)

# Glucose
glucose_bins = [0, 99, 125, 200]
glucose_labels = ['Normal', 'Pre-diabetic', 'Diabetic']
df['glucose_group'] = pd.cut(df['Glucose'], bins=glucose_bins,
labels=glucose_labels)

# Insulin
insulin_bins = [0, 15, 100, 200, 1000]
insulin_labels = ['Very Low', 'Low', 'Moderate', 'High']
df['insulin_group'] = pd.cut(df['Insulin'], bins=insulin_bins,
labels=insulin_labels)

# Function to summarize prevalence
```

```
def prevalence_by_group(var):
    return df.groupby(var, observed=True)['Outcome'].mean().round(3) *
100

# Calculate prevalence across all groups
age_prev = prevalence_by_group('age_group')
bmi_prev = prevalence_by_group('bmi_group')
preg_prev = prevalence_by_group('preg_group')
glucose_prev = prevalence_by_group('glucose_group')
insulin_prev = prevalence_by_group('insulin_group')

print(age_prev)
print(bmi_prev)
print(preg_prev)
print(glucose_prev)
print(insulin_prev)
```

OUTPUT

Group	Category	Prevalence (%)
Age	20–29	21.2
	30–39	46.1
	40–49	55.1
	50–59	59.6
	60+	28.1
BMI	Underweight	0.0
	Normal	6.9
	Overweight	22.3
	Obese	46.6
	Severely Obese	46.3
Pregnancies	0	34.2
	1	21.5

	2	18.4
	3	36.0
	4+	45.1
Glucose	Normal	7.3
	Pre-Diabetic	27.7
	Diabetic	59.3
Insulin	Very Low	50.0
	Low	12.1
	Moderate	41.1
	High	54.1
Diabetes Pedigree Function*	Q1	25.5
	Q2	33.3
	Q3	32.3
	Q4	48.4
* There is no categorization that is clear for DPF, so it is binned into quartiles		

REFLECTION #3B

Where is diabetes most common?

Diabetes is most common among the following groups, with all showing prevalence above 45-55%, which is significantly higher than the population average of ~35%:

- Individuals aged 40-59
- Those in the obese and severely obese BMI categories
- People with 4 or more pregnancies
- Individuals with glucose levels in the diabetic range
- Those with high insulin levels
- People in the highest quartile of Diabetes Pedigree Function (DPF)

One surprise is that the lowest insulin group has a relatively high prevalence at 50%, but this may be due to it capturing those with missing insulin values which were coded as “0”.

Which grouping method worked best?

Groupings based on clinical cutoffs such as BMI or glucose, or known epidemiological categories such as age or pregnancies, were especially useful for interpretation and clarity. Using quartiles worked best for DPF as well, as there are no universal cutoffs for that variable.

<u>Wrap-Up</u>

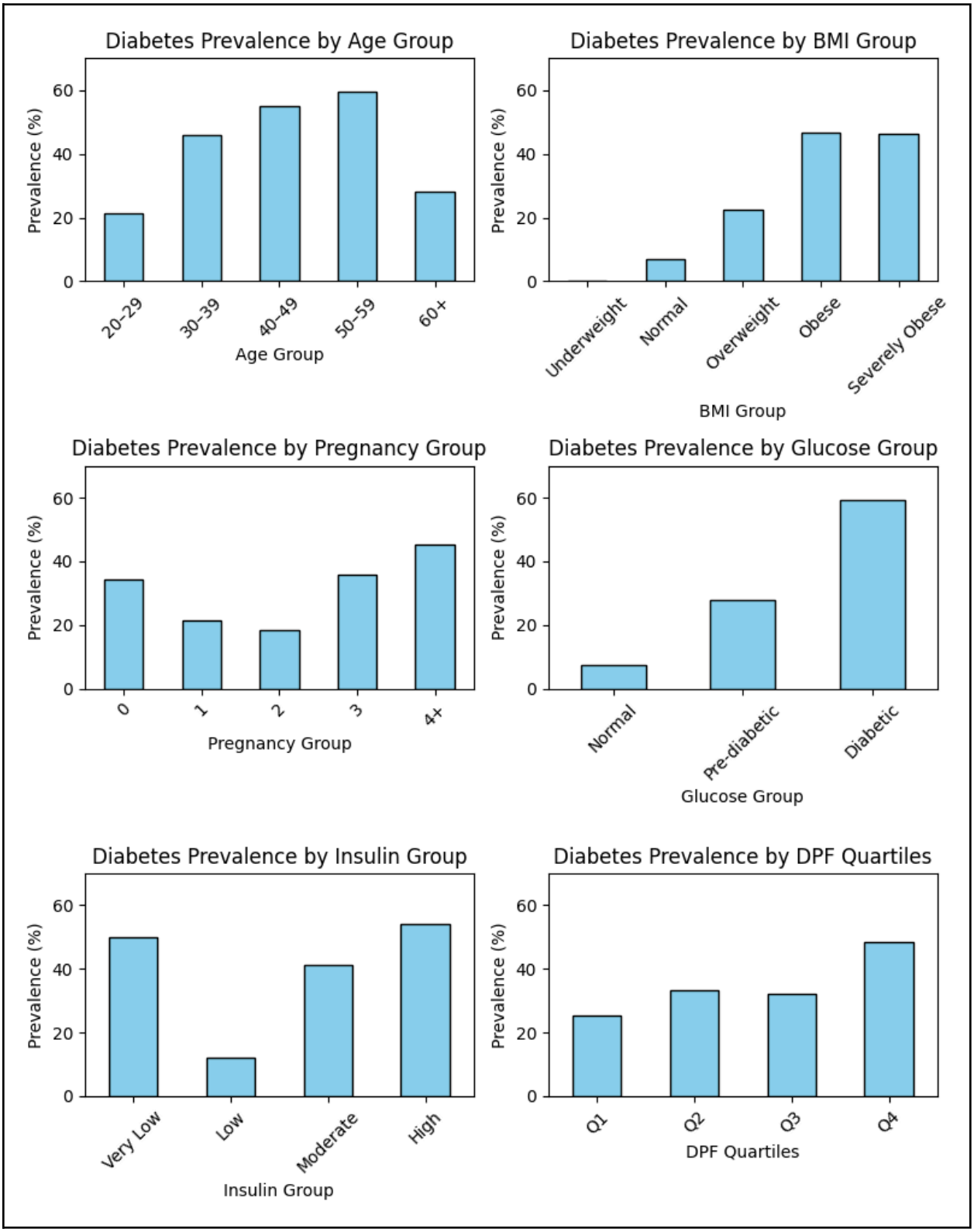
Diabetes prevalence is not evenly distributed in the population, with some groups having noticeably higher rates. Glucose, BMI, pregnancies, and DPF all show strong gradient-based relationships with diabetes status, and these patterns will help with risk modeling.

STEP 3C. VISUALIZING BY GROUP

CODE

```
# STEP 3C

import matplotlib.pyplot as plt

# Group prevalence series in a dictionary
grouped_prevalences = {
    'Age Group': age_prev,
    'BMI Group': bmi_prev,
    'Pregnancy Group': preg_prev,
    'Glucose Group': glucose_prev,
    'Insulin Group': insulin_prev,
    'DPF Quartiles': dpf_prev
}

# Size of entire figure (w x h in inches)
plt.figure(figsize=(8, 10))

# Loop through the dictionary and create subplot per variable
for i, (label, series) in enumerate(grouped_prevalences.items(), 1):
    # Create a subplot: 3 rows, 2 columns, position i
    plt.subplot(3, 2, i)
    series.plot(kind='bar', color='skyblue', edgecolor='black')
    plt.title(f'Diabetes Prevalence by {label}')
    plt.ylabel('Prevalence (%)')
    plt.xlabel(label)
    # Standardize y-axis to 0-70% for consistency
    plt.ylim(0, 70)
    plt.xticks(rotation=45)

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

OUTPUT


Diabetes Prevalence by Age Group
Prevalence (%)
0
20
40
60
20–29
30–39
40–49
50–59
60+
Age Group
Diabetes Prevalence by BMI Group
Prevalence (%)
0
20
40
60
Underweight
Normal
Overweight
Obese
Severely Obese
BMI Group
Diabetes Prevalence by Pregnancy Group
Prevalence (%)
0
20
40
60
0
1
2
3
4+
Pregnancy Group
Diabetes Prevalence by Glucose Group
Prevalence (%)
0
20
40
60
Normal
Pre-diabetic
Diabetic
Glucose Group
Diabetes Prevalence by Insulin Group
Prevalence (%)
0
20
40
60
Very Low
Low
Moderate
High
Insulin Group
Diabetes Prevalence by DPF Quartiles
Prevalence (%)
0
20
40
60
Q1
Q2
Q3
Q4
DPF Quartiles

REFLECTION #3C

What patterns jump out visually?

There is a sharp spike in prevalence in the diabetic glucose range, obese/severely obese groups and those aged 40-59. Groups with normal glucose, underweight and normal BMI and low insulin show very low risk.

Wrap-Up

Each bar plot supports the earlier numeric findings while making the key risk patterns more recognizable.

STEP 4. RISK RATIOS AND ODDS RATIOS

In this step, risk ratios and odds ratios will be calculated for specific group comparisons. These measures quantify how much more likely one group is than another to have diabetes, which help assess the strength of the association between the exposures and outcome.

Quick Refresh

- Risk Ratios (RR) compares the probability of an outcome between two groups, such as exposed vs. unexposed. RR's are more intuitive in interpretation.
    - RR = 1 is no difference, RR > 1 is increased risk
- Odds Ratios (OR) compares the odds of an outcome between two groups. ORs are often used in logistic regression and show up more in modeling.
    - OR > 1 is increased odds, and is a bit more extreme than RR

2*2 Contingency table

	Cases	Controls	Total
Exposed	a	b	a+b
Unexposed	c	d	c+d
Total	a+c	b+d	a+b+c+d

OR = (a/b)/(c/d)

RR = (a/a+b)/(c/c+d)

CODE

```
# RR and OR for two groups
def calc_rr_or(df, group_var, ref_group, comp_group):
    # Get counts
    exposed = df[df[group_var] == comp_group]['Outcome']
```

```
    unexposed = df[df[group_var] == ref_group]['Outcome']

    # Calculate risk
    risk_exp = exposed.mean()
    risk_unexp = unexposed.mean()
    rr = risk_exp / risk_unexp

    # Calculate odds
    odds_exp = risk_exp / (1 - risk_exp)
    odds_unexp = risk_unexp / (1 - risk_unexp)
    or_val = odds_exp / odds_unexp

    # Round
    return round(rr, 2), round(or_val, 2)

# Groups for each major variable
rr_age, or_age = calc_rr_or(df, 'age_group', '20-29', '40-49')
rr_bmi, or_bmi = calc_rr_or(df, 'bmi_group', 'Normal', 'Obese')
rr_glucose, or_glucose = calc_rr_or(df, 'glucose_group', 'Normal',
'Diabetic')
rr_preg, or_preg = calc_rr_or(df, 'preg_group', '1', '4+')

# Display results
print("Risk Ratios and Odds Ratios:")
print(f"Age (40-49 vs 20-29): RR = {rr_age}, OR = {or_age}")
print(f"BMI (Obese vs Normal): RR = {rr_bmi}, OR = {or_bmi}")
print(f"Glucose (Diabetic vs Normal): RR = {rr_glucose}, OR =
{or_glucose}")
print(f"Pregnancies (4+ vs 1): RR = {rr_preg}, OR = {or_preg}")
```

OUTPUT

```
Risk Ratios and Odds Ratios:
Age (40-49 vs 20-29): RR = 2.6, OR = 4.56
BMI (Obese vs Normal): RR = 6.79, OR = 11.85
Glucose (Diabetic vs Normal): RR = 8.13, OR = 18.49
Pregnancies (4+ vs 1): RR = 2.1, OR = 3.0
```

REFLECTION #4

What do the RR and OR show?

Women aged 40–49 are 2.6 times more likely to have diabetes than those aged 20–29. The likelihood of diabetes is 6.8 times higher in women classified as obese compared to those with a normal BMI. Among those in the diabetic-range glucose group, diabetes is 8.1 times more common than in individuals with normal glucose levels. Lastly, women with 4 or more pregnancies are 2.1 times as likely to have diabetes as those who have had just one pregnancy.

Wrap-Up

The strongest predictors based on risk ratio and odds ratios are glucose and BMI, which mirrors what was found with prevalence.

STEP 5. CRUDE LOGISTIC REGRESSION

In this step, simple (crude) logistic regression models are run to see how each major variable - age, BMI, glucose, and pregnancies - is individually associated with diabetes. These models will show how the odds of the outcome change with each unit increase in the variable, without adjusting for any other factors.

Quick Refresh:

- Logistic regression estimates the probability of a binary outcome
    - It outputs ORs for each predictor along with p-values showing whether the association is statistically significant

CODE

```
import statsmodels.api as sm
import numpy as np

# Define outcome
y = df['Outcome']

# Create a function for crude logistic regression
def run_crude_logit(predictor):
    X = sm.add_constant(df[[predictor]])
    model = sm.Logit(y, X).fit(disp=False)
    or_val = np.exp(model.params.iloc[1])  # Future-proof fix
    p_val = model.pvalues.iloc[1]          # Future-proof fix
    return round(or_val, 2), round(p_val, 5)

# Run for each predictor
```

```
crude_age = run_crude_logit('Age')
crude_bmi = run_crude_logit('BMI')
crude_glucose = run_crude_logit('Glucose')
crude_preg = run_crude_logit('Pregnancies')

# Print results
print("Age:         OR =", crude_age[0], "p =", crude_age[1])
print("BMI:         OR =", crude_bmi[0], "p =", crude_bmi[1])
print("Glucose:     OR =", crude_glucose[0], "p =", crude_glucose[1])
print("Pregnancies:OR =", crude_preg[0], "p =", crude_preg[1])
```

OUTPUT

```
Age:         OR = 1.04 p = 0.0
BMI:         OR = 1.1 p = 0.0
Glucose:     OR = 1.04 p = 0.0
Pregnancies:OR = 1.15 p = 0.0
```

REFLECTION #5

What do these odds ratios mean? Are these predictors statistically significant?

For each additional year of age, odds of diabetes increase by 5%. Increasing BMI by 1 unit increases the odds of the outcome by 10%. Each additional 1 unit increase of glucose increases the odds by 4%. And lastly, for each additional pregnancy, odds increase by 13%.

All of these predictors have a p-value of <0.05, meaning that they show statistically significant associations with diabetes in crude models.

Wrap-Up

All four variables (age, BMI, glucose, and pregnancies) are significantly associated with diabetes risk, which supports putting them in adjusted models to check for confounding and combined predictive power.

STEP 6. ADJUSTED LOGISTIC REGRESSION

Quick Refresh

- Adjusted logistic regression (multiple predictors) gives
  - Adjusted odds ratios (AORs) which is how the odds of diabetes changes with each variable holding the other(s) constant

CODE

```
# Build multivariate model
X_multi = sm.add_constant(df[['BMI', 'Glucose', 'Pregnancies']])
model_multi = sm.Logit(y, X_multi).fit()

# Summarize outputs
# Summarize model output
params = model_multi.params
pvals = model_multi.pvalues
or_vals = params.apply(lambda x: round(np.exp(x), 2))

# Create a table
summary_table = pd.DataFrame({
    'OR': or_vals,
    'p-value': pvals.round(5)
})

print(summary_table)
```

OUTPUT

```
Optimization terminated successfully.
         Current function value: 0.484456
         Iterations 6
                OR  p-value
const         0.00      0.0
BMI           1.08      0.0
Glucose       1.03      0.0
Pregnancies   1.15      0.0
```

REFLECTION #6

How do these adjusted odds ratios compare to the crude ones? Any signs of confounding?

ORs are slightly lower than in the crude modes, which suggests some overlap in predictive power, but each variable does still hold significance.

There are signs of confounding as each variable's effect was slightly reduced when others were added, which could mean there was some shared variance, but all remain important predictors.

How would you explain this model to a policymaker?

Women with higher glucose, BMI, or pregnancy counts are more likely to have diabetes - even when adjusting for the other factors. Each factor contributes independently to the risk.

Wrap-Up

Glucose, BMI, and pregnancies are all statistically significant independent predictors of diabetes. The odds of diabetes increases 3% per unit increase in glucose, 8% per unit increase in BMI< and 15% per additional pregnancy.

STEP 7. MODEL PERFORMANCE

In this step, the adjusted logistic regression model is used to make predictions and evaluate how well it classifies cases. Accuracy, sensitivity, and specificity will be calculated, and a confusion matrix used to understand performance.

Quick Refresh:

- Threshold is 0.5, with above being diabetic and below being non-diabetic
- Confusion matrix shows true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).
    - From this the following can be calculated:
        - Accuracy = (TP + TN) / total
        - Sensitivity (Recall) = TP / (TP + FN)
        - Specificity = TN / (TN + FP)

CODE

```
from sklearn.metrics import confusion_matrix, classification_report

# Get predicted probabilities
df['predicted_prob'] = model_multi.predict(X_multi)

# Classify predicitons using threshold
df['predicted_outcome'] = df['predicted_prob'].apply(lambda x: 1 if x >
0.5 else 0)

# Confusion matrix
conf_matrix = confusion_matrix(df['Outcome'], df['predicted_outcome'])
TN, FP, FN, TP = conf_matrix.ravel()

# Classification report
report = classification_report(df['Outcome'], df['predicted_outcome'],
output_dict=True)
print(pd.DataFrame(report).T)
```

```
# Print confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
# formatted view
print(f"\nTN: {TN}, FP: {FP}, FN: {FN}, TP: {TP}")
```

OUTPUT

```
              precision    recall  f1-score     support
0              0.790235  0.874000  0.830009  500.000000
1              0.706977  0.567164  0.629400  268.000000
accuracy       0.766927  0.766927  0.766927    0.766927
macro avg      0.748606  0.720582  0.729705  768.000000
weighted avg   0.761181  0.766927  0.760005  768.000000
Confusion Matrix:
[[437  63]
 [116 152]]

TN: 437, FP: 63, FN: 116, TP: 152
```

REFLECTION #7

How accurate is your model? Any concerns?

The model has 77% overall accuracy, with strong specificity (87%) but only moderate sensitivity (57%). This means it correctly identifies most non-diabetic individuals but misses a substantial portion of true diabetes cases. This could pose a risk in screening programs, where missing true cases is a serious concern.

What would you recommend?

This model could be useful in clinical settings for risk stratification—where it's important to rule out low-risk individuals—but it’s not suitable for broad public health screening unless sensitivity improves (e.g., by adjusting the classification threshold).

Wrap-Up

The model performs well at identifying non-diabetics but less well at catching true diabetics - visible in the confusion matrix and supported by precision/recall metrics.

STEP 8. ROC CURVE AND THRESHOLD ADJUSTMENT

In this step, a Receiver Operating Characteristic (ROC) is generated and the Area Under the Curve (AUC) is calculated to evaluate the model’s ability to distinguish between diabetic and non-diabetic individuals across all possible thresholds. Performance at different thresholds (e.g., 0.4 vs. 0.5) will also be compared to see how sensitivity and specificity shift.

Quick Refresh:

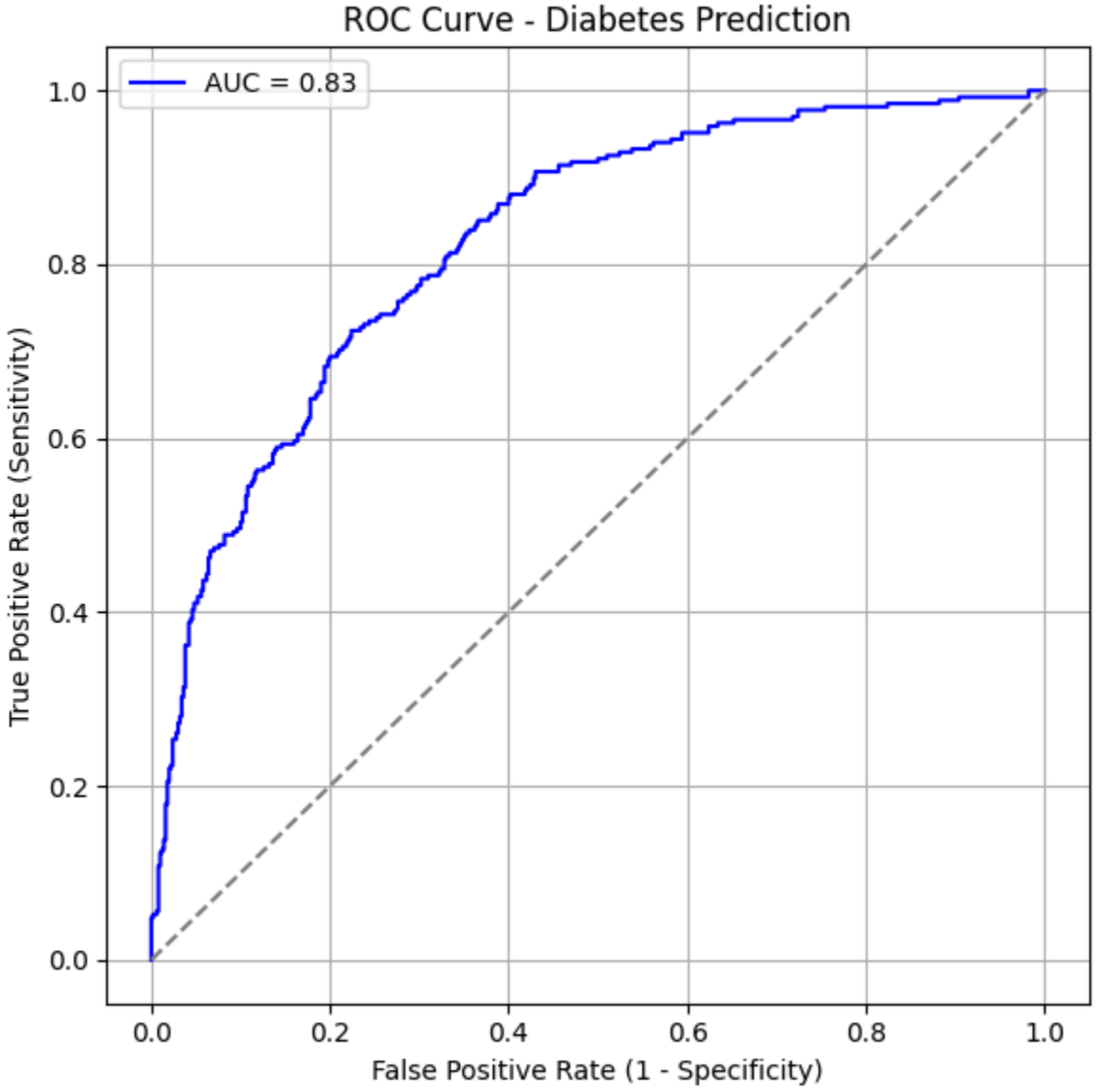
- The ROC Curve is a graph of sensitivity (True Positive Rate) vs. 1-specificity (False Positive Rate)
- The AUC Score is the probability the model ranks a random positive case higher than a negative one

CODE

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# ROC and AUC
fpr, tpr, thresholds = roc_curve(df['Outcome'], df['predicted_prob'])
auc_score = roc_auc_score(df['Outcome'], df['predicted_prob'])

# Plot ROC Curve
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}", color='blue')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.title('ROC Curve - Diabetes Prediction')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Adjust threshold to 0.4 and compare classification
df['predicted_0.4'] = df['predicted_prob'].apply(lambda x: 1 if x > 0.4
else 0)
conf_matrix_0_4 = confusion_matrix(df['Outcome'], df['predicted_0.4'])
TN2, FP2, FN2, TP2 = conf_matrix_0_4.ravel()
print(f"Threshold 0.4 — TN: {TN2}, FP: {FP2}, FN: {FN2}, TP: {TP2}")

# Sensitivity and Specificity at threshold 0.4
sensitivity_0_4 = TP2 / (TP2 + FN2)
specificity_0_4 = TN2 / (TN2 + FP2)

print(f"Sensitivity: {sensitivity_0_4:.2f}")
```

```
print(f"Specificity: {specificity_0_4:.2f}")
```

OUTPUT




```
Threshold 0.4 — TN: 409, FP: 91, FN: 95, TP: 173
Sensitivity: 0.65
Specificity: 0.82
```

REFLECTION #8

How did sensitivity and specificity change at a lower threshold?

Sensitivity increased from 57% to 64%, meaning more diabetic cases were correctly identified. However, specificity decreased from 87% to 82%, meaning more non-diabetic individuals were misclassified.

Is 0.4 a better threshold for screening? Why or why not?

Yes, for screening purposes, the increased sensitivity is valuable—it helps catch more real cases. However, the drop in specificity could result in more false positives, which might lead to unnecessary anxiety, referrals, or testing, negatively impacting the women.

What does the ROC curve and AUC tell us about the model overall?

With an AUC of 0,83, the model has good discrimination which means it is reliably ranking diabetic cases above non-cases, while the curve shows strong performance across thresholds.

Wrap-Up

The model's AUC of 0.83 suggests good overall diagnostic accuracy. Lowering the threshold can improve sensitivity for screening, but increases false positives.

STEP 9. CHECKING FOR CONFOUNDING

In this step, two-variable logistic regression models will be built to check for confounding.

Quick Refresh:

- A confounder is a variable that is associated with both the outcome and the predictor independently, but is not on the causal pathway.

CODE

```
# Logistic regression summary for selected variables
def run_logit_and_report(vars_list):
    X = sm.add_constant(df[vars_list])
    model = sm.Logit(y, X).fit(disp=False)
    or_vals = model.params.apply(lambda x: round(np.exp(x), 2))
    p_vals = model.pvalues.round(5)
    return pd.DataFrame({'OR': or_vals, 'p-value': p_vals})

# BMI + Age
bmi_age = run_logit_and_report(['BMI', 'Age'])

# BMI + Pregnancies
bmi_preg = run_logit_and_report(['BMI', 'Pregnancies'])

# BMI + Glucose
```

```
bmi_gluc = run_logit_and_report(['BMI', 'Glucose'])

# Glucose + Pregnancies
gluc_preg = run_logit_and_report(['Glucose', 'Pregnancies'])

print("\nBMI + Age:")
print(bmi_age)
print("\nBMI + Pregnancies:")
print(bmi_preg)
print("\nBMI + Glucose:")
print(bmi_gluc)
print("\nGlucose + Pregnancies:")
print(gluc_preg)
```

OUTPUT

Model	Variable	Crude OR	Adjusted OR	p-value
BMI + Age	BMI	1.10	1.10	<0.001
	Age	1.05	1.05	<0.001
BMI + Pregnancies	BMI	1.10	1.10	<0.001
	Pregnancies	1.13	1.16	<0.001
BMI + Glucose	BMI	1.10	1.08	<0.001
	Glucose	1.04	1.04	<0.001
Glucose + Pregnancies	Glucose	1.04	1.04	<0.001
	Pregnancies	1.13	1.13	<0.001

REFLECTION #9

Was there evidence of confounding? Did all predictors remain statistically significant?

In the BMI + Glucose model, BMI's OR dropped from 1.10 (crude) to 1.08 (adjusted). This suggests some shared predictive power—likely because glucose and BMI are both strong, overlapping indicators of metabolic health. Other ORs remained identical or nearly unchanged, suggesting little to no confounding.

All variables had p-values < 0.001 across models, meaning they're strong independent predictors of diabetes.

Wrap-Up

BMI, Glucose, Age, and Pregnancies are all statistically significant predictors, both alone and when adjusted for each other. The slight drop in BMI's OR when adjusting for Glucose suggests partial confounding, but not enough to dismiss BMI as a meaningful independent predictor.

STEP 10. FINAL MODEL SUMMARY

In this step, the logistic regression model is finalized using three key predictors: BMI, Glucose, and Pregnancies. This model was selected based on previous steps showing each variable's independent predictive power and minimal confounding between them.

CODE

```
# Logistic regression summary for selected variables
def run_logit_and_report(vars_list):
    X = sm.add_constant(df[vars_list])
    model = sm.Logit(y, X).fit(disp=False)
    or_vals = model.params.apply(lambda x: round(np.exp(x), 2))
    p_vals = model.pvalues.round(5)
    return pd.DataFrame({'OR': or_vals, 'p-value': p_vals})

# BMI + glucose + pregnancies
bmi_gluc_preg = run_logit_and_report(['BMI', 'Glucose', 'Pregnancies'])

print("\nBMI + Glucose + Pregnancies:")
print(bmi_gluc_preg)
```

OUTPUT

Predictor	**OR**	**p-value**
Constant	0.00	<0.001
BMI	1.08	<0.001
Glucose	1.03	<0.001
Pregnancies	1.15	<0.001

REFLECTION #10

What does this model tell us? Are the predictors still statistically significant?

For each 1-unit increase in BMI, the odds of having diabetes increase by 8%, controlling for glucose and pregnancies. For each 1-unit increase in Glucose, odds increase by 3%,

adjusting for BMI and pregnancies. Each additional pregnancy increases odds by 15%, holding BMI and glucose constant.

All three predictors remain highly significant ($p < 0.001$), meaning their associations with diabetes are unlikely due to chance, even after adjusting for each other.

Wrap-Up

The final model shows that BMI, Glucose, and Pregnancies are statistically significant, independent predictors of diabetes in this population. This multivariable approach strengthens the evidence that diabetes risk is multifactorial and must be evaluated holistically—not in isolation.

STEP 11. OPTIONAL MODEL TESTING

In this step, the user is allowed to enter hypothetical patient values for BMI, Glucose, and Pregnancies, and use the logistic regression model to predict their risk of diabetes.

CODE

```
# Function 1: Test a single patient
def test_patient_risk(bmi, glucose, pregnancies):
    """
    Estimate the probability of diabetes for one individual.

    Inputs:
    - bmi: Body Mass Index (e.g., 27 for overweight)
    - glucose: Fasting glucose level (e.g., 120 for pre-diabetic)
    - pregnancies: Number of pregnancies (e.g., 1)

    Output:
    - Predicted diabetes probability as a formatted string
    """
    new_patient = pd.DataFrame([{
        'BMI': bmi,
        'Glucose': glucose,
        'Pregnancies': pregnancies
    }])

    new_patient = sm.add_constant(new_patient, has_constant='add')
    predicted_risk = model_multi.predict(new_patient)[0]
    return f"Predicted Probability of Diabetes: {predicted_risk:.1%}"
```

```
# Examples:
print(test_patient_risk(27, 120, 1))
print(test_patient_risk(35, 160, 4))

# Function 2: Test multiple patients at once
def test_multiple_patients(patients_list):
    """
    Estimate diabetes risk for multiple individuals.

    Input:
    - patients_list: A list of dictionaries with 'BMI', 'Glucose', and
'Pregnancies' values

    Output:
    - A DataFrame showing each profile and predicted probability
    """
    patients_df = pd.DataFrame(patients_list)
    patients_df = sm.add_constant(patients_df, has_constant='add') #
adds intercept
    patients_df['Predicted Risk (%)'] =
model_multi.predict(patients_df).apply(lambda x: round(x * 100, 1)) #
adds each patient as a new column
    return patients_df.drop(columns='const') # cleans up output

# Examples:
example_patients = [
    {'BMI': 22, 'Glucose': 95,  'Pregnancies': 0},
    {'BMI': 30, 'Glucose': 130, 'Pregnancies': 2},
    {'BMI': 36, 'Glucose': 170, 'Pregnancies': 5}
]

print(test_multiple_patients(example_patients))
```

OUTPUT

```
Predicted Probability of Diabetes: 15.6%
Predicted Probability of Diabetes: 67.8%

   BMI  Glucose  Pregnancies  Predicted Risk (%)
0   22       95            0                 4.4
1   30      130            2                27.6
2   36      170            5                78.7
```