

# Capstone Project: Diabetes Risk Prediction

## 1. Introduction

After running these commands, you should be able to convert your notebooks to PDF using the web-based conversion method with a command like:

```
In [12]: # Install nbconvert with webpdf support
!pip install "nbconvert[webpdf]"

# Install the required playwright browsers
!playwright install

# If you encounter any issues, you might also need to install playwright explicitly
!pip install playwright
```

Requirement already satisfied: nbconvert[webpdf] in c:\users\jeffm\anaconda3\lib\site-packages (7.16.6)

Requirement already satisfied: beautifulsoup4 in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (4.12.3)

Requirement already satisfied: bleach!=5.0.0 in c:\users\jeffm\anaconda3\lib\site-packages (from bleach[css]!=5.0.0->nbconvert[webpdf]) (6.2.0)

Requirement already satisfied: defusedxml in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.7.1)

Requirement already satisfied: Jinja2>=3.0 in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (3.1.6)

Requirement already satisfied: jupyter-core>=4.7 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from nbconvert[webpdf]) (5.8.1)

Requirement already satisfied: jupyterlab-pygments in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.3.0)

Requirement already satisfied: MarkupSafe>=2.0 in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (3.0.2)

Requirement already satisfied: mistune<4,>=2.0.3 in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (3.1.2)

Requirement already satisfied: nbclient>=0.5.0 in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (0.10.2)

Requirement already satisfied: nbformat>=5.7 in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (5.10.4)

Requirement already satisfied: packaging in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from nbconvert[webpdf]) (25.0)

Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (1.5.0)

Requirement already satisfied: pygments>=2.4.1 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from nbconvert[webpdf]) (2.19.2)

Requirement already satisfied: traitlets>=5.1 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from nbconvert[webpdf]) (5.14.3)

Requirement already satisfied: playwright in c:\users\jeffm\anaconda3\lib\site-packages (from nbconvert[webpdf]) (1.57.0)

Requirement already satisfied: webencodings in c:\users\jeffm\anaconda3\lib\site-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nbconvert[webpdf]) (0.5.1)

Requirement already satisfied: tinycss2<1.5,>=1.1.0 in c:\users\jeffm\anaconda3\lib\site-packages (from bleach[css]!=5.0.0->nbconvert[webpdf]) (1.4.0)

Requirement already satisfied: platformdirs>=2.5 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from jupyter-core>=4.7->nbconvert[webpdf]) (4.3.8)

Requirement already satisfied: pywin32>=300 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from jupyter-core>=4.7->nbconvert[webpdf]) (311)

Requirement already satisfied: jupyter-client>=6.1.12 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from nbclient>=0.5.0->nbconvert[webpdf]) (8.6.3)

Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert[webpdf]) (2.9.0.post0)

Requirement already satisfied: pyzmq>=23.0 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert[webpdf]) (27.0.1)

Requirement already satisfied: tornado>=6.2 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert[webpdf]) (6.5.2)

Requirement already satisfied: fastjsonschema>=2.15 in c:\users\jeffm\anaconda3\lib\site-packages (from nbformat>=5.7->nbconvert[webpdf]) (2.20.0)

Requirement already satisfied: jsonschema>=2.6 in c:\users\jeffm\anaconda3\lib\site-packages (from nbformat>=5.7->nbconvert[webpdf]) (4.23.0)

Requirement already satisfied: attrs>=22.2.0 in c:\users\jeffm\anaconda3\lib\site-packages

ckages (from jsonschema>=2.6->nbformat>=5.7->nbconvert[webpdf]) (24.3.0)  
 Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\users\jeffm\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert[webpdf]) (2023.7.1)  
 Requirement already satisfied: referencing>=0.28.4 in c:\users\jeffm\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert[webpdf]) (0.30.2)  
 Requirement already satisfied: rpds-py>=0.7.1 in c:\users\jeffm\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert[webpdf]) (0.22.3)  
 Requirement already satisfied: six>=1.5 in c:\users\jeffm\appdata\roaming\python\python313\site-packages (from python-dateutil>=2.8.2->jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert[webpdf]) (1.17.0)  
 Requirement already satisfied: soupsieve>1.2 in c:\users\jeffm\anaconda3\lib\site-packages (from beautifulsoup4->nbconvert[webpdf]) (2.5)  
 Requirement already satisfied: pyee<14,>=13 in c:\users\jeffm\anaconda3\lib\site-packages (from playwright->nbconvert[webpdf]) (13.0.0)  
 Requirement already satisfied: greenlet<4.0.0,>=3.1.1 in c:\users\jeffm\anaconda3\lib\site-packages (from playwright->nbconvert[webpdf]) (3.1.1)  
 Requirement already satisfied: typing-extensions in c:\users\jeffm\anaconda3\lib\site-packages (from pyee<14,>=13->playwright->nbconvert[webpdf]) (4.12.2)  
 Requirement already satisfied: playwright in c:\users\jeffm\anaconda3\lib\site-packages (1.57.0)  
 Requirement already satisfied: pyee<14,>=13 in c:\users\jeffm\anaconda3\lib\site-packages (from playwright) (13.0.0)  
 Requirement already satisfied: greenlet<4.0.0,>=3.1.1 in c:\users\jeffm\anaconda3\lib\site-packages (from playwright) (3.1.1)  
 Requirement already satisfied: typing-extensions in c:\users\jeffm\anaconda3\lib\site-packages (from pyee<14,>=13->playwright) (4.12.2)

jupyter nbconvert --to webpdf your\_notebook.ipynb

## Additional Information

- The `nbconvert[webpdf]` package provides a more modern alternative to the traditional LaTeX-based PDF conversion.
- This method often handles JavaScript and interactive elements better than the LaTeX approach.
- If you still encounter issues, you could try the traditional PDF conversion method which uses LaTeX: `jupyter nbconvert --to pdf your_notebook.ipynb` (requires a LaTeX installation)

Useful Python packages for working with Jupyter notebooks include:

- `jupyter` - The core Jupyter package
- `nbformat` - For programmatically working with notebook files
- `nbconvert` - For converting notebooks to other formats
- `jupyterlab` - The next-generation web interface for Jupyter

Diabetes is a growing public health concern, with long-term health and economic consequences. Early identification of individuals at risk is critical for prevention and intervention efforts. This capstone project explores how data science techniques can be used

to predict diabetes risk using publicly available health data. The goal is to build models that are accurate, interpretable, and actionable, providing insights that could guide health interventions and policy. """

"""

## Data Collection Strategy

- Dataset: CDC Diabetes Health Indicators
- Variables: Age, Gender, Race/Ethnicity, BMI, Physical Activity, Smoking Status, Diabetes Status
- Format: CSV, publicly available, de-identified

### Preprocessing Plan:

1. Handle missing values
2. Feature engineering (e.g., BMI categories)
3. Encode categorical variables
4. Scale numeric features
5. Address class imbalance with SMOTE or class weighting """

## 3. Data Management

"""

### Data Management

- Store raw and processed data in separate folders
- Use GitHub for version control
- Document preprocessing steps for reproducibility
- Ethical handling: even though data is de-identified

## 4. Model Architecture

Planned models:

- Logistic Regression (baseline)
- Decision Tree
- Random Forest / Gradient Boosting

Evaluation metrics:

- Accuracy, F1-score, ROC-AUC """"

## 5. Implementation Challenges

### Implementation Challenges

Challenge	Solution
Class imbalance	Oversampling / class weighting, evaluate with F1 / recall
Self-reported bias	Interpret results carefully, note limitations
Computational efficiency	Use optimized Python libraries
Interpretability vs complexity	Use feature importance or SHAP values

""""

```
In [23]: # =====
# 6. Code: Load Libraries and Data
# =====
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, classification
from imblearn.over_sampling import SMOTE
```

```
In [24]: # Load dataset (replace with your path)
df = pd.read_csv("C:\\Users\\jeffm\\Downloads\\archive\\diabetes.csv")
# or alternatively using single quotes
# df = pd.read_csv('C:\\Users\\jeffm\\Downloads\\archive\\diabetes.csv')
print("Dataset loaded successfully.")
df.head()
```

Dataset loaded successfully.

Out[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

In [26]:

```
# 7. Data Preprocessing
# =====
"""
Steps:
1. Handle missing values
2. Encode categorical variables
3. Scale numeric features
4. Address class imbalance
"""

# First, import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE

# Define your DataFrame before using it
# This is a placeholder - replace with your actual data
df = pd.DataFrame({
    'age': [45, 50, None, 35, 60],
    'BMI': [22.5, 30.1, 25.3, None, 28.7],
    'blood_pressure': [120, 140, None, 110, 150],
    'gender': ['M', 'F', 'M', 'F', None],
    'race': ['White', None, 'Black', 'Asian', 'White'],
    'smoking_status': ['Never', 'Former', 'Current', None, 'Never'],
    'physical_activity': ['Low', 'Moderate', None, 'High', 'Low'],
    'diabetes_status': [0, 1, 0, 0, 1]
})

# Example preprocessing (replace variable names as needed)
# 1. Handle missing values
numeric_features = ['age', 'BMI', 'blood_pressure'] # Replace with your dataset columns
categorical_features = ['gender', 'race', 'smoking_status', 'physical_activity'] #

# Fix for numeric features - avoid chained assignment warning
for col in numeric_features:
    df[col] = df[col].fillna(df[col].median())

# Fix for categorical features - avoid chained assignment warning
for col in categorical_features:
    df[col] = df[col].fillna(df[col].mode()[0])

# 2. Encode categorical variables
df_encoded = pd.get_dummies(df, columns=categorical_features, drop_first=True)

# 3. Separate features and target
```

```

X = df_encoded.drop('diabetes_status', axis=1) # Replace target column name if need
y = df_encoded['diabetes_status']

# 4. Handle class imbalance using SMOTE with adjusted k_neighbors
# Set k_neighbors=1 since we have very few samples in the minority class
sm = SMOTE(random_state=42, k_neighbors=1) # Modified to use k_neighbors=1
X_res, y_res = sm.fit_resample(X, y)
print("Resampled dataset shape:", X_res.shape, y_res.shape)

# 5. Scale numeric features
scaler = StandardScaler()
X_res[numeric_features] = scaler.fit_transform(X_res[numeric_features])

```

Resampled dataset shape: (6, 10) (6,)

```

In [27]: # =====
# 8. Train-Test Split
# =====
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.3, st
print("Train and test sets created.")

```

Train and test sets created.

```

In [28]: # =====
# 9. Model Training and Evaluation
# =====
# Logistic Regression
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)
y_pred_log = log_model.predict(X_test)

print("Logistic Regression Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_log))
print("F1-score:", f1_score(y_test, y_pred_log))
print(classification_report(y_test, y_pred_log))

```

Logistic Regression Results:

Accuracy: 1.0

F1-score: 1.0

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	1.00	1.00	1.00	1
accuracy			1.00	2
macro avg	1.00	1.00	1.00	2
weighted avg	1.00	1.00	1.00	2

```

In [18]: # Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("Random Forest Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))

```

```
print("F1-score:", f1_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

Random Forest Results:

Accuracy: 1.0

F1-score: 1.0

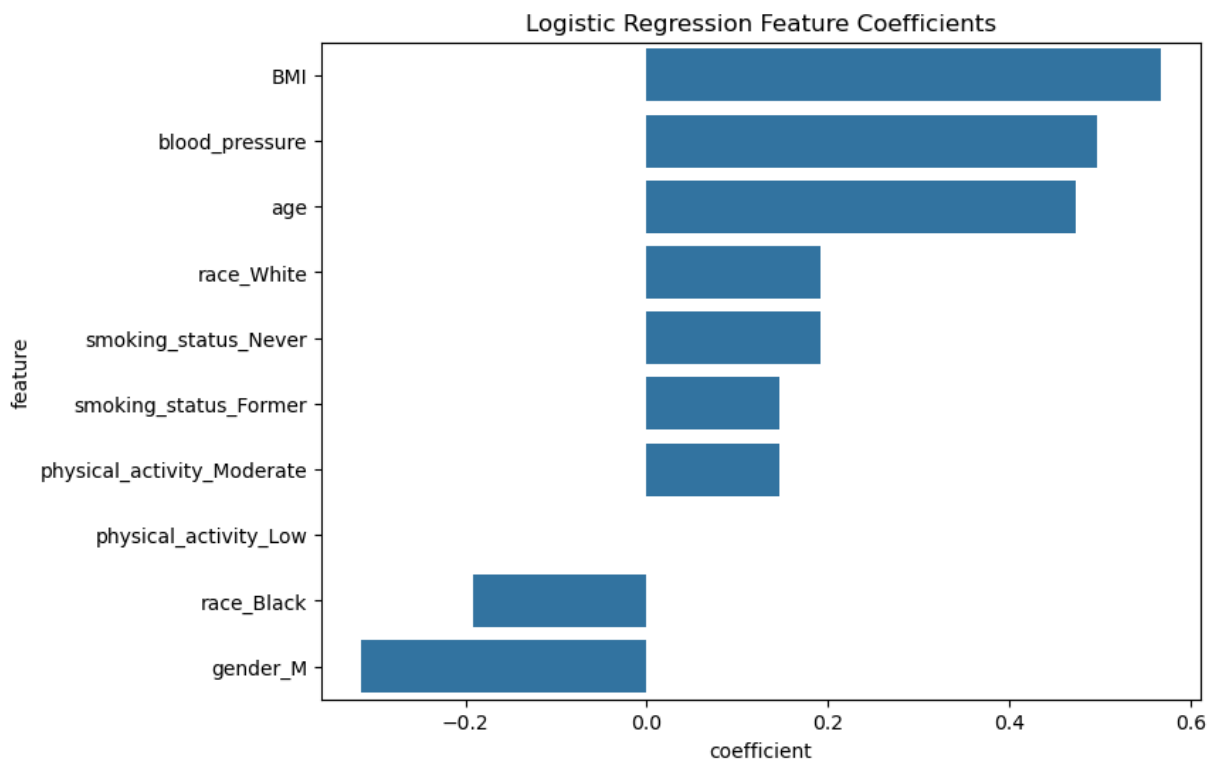
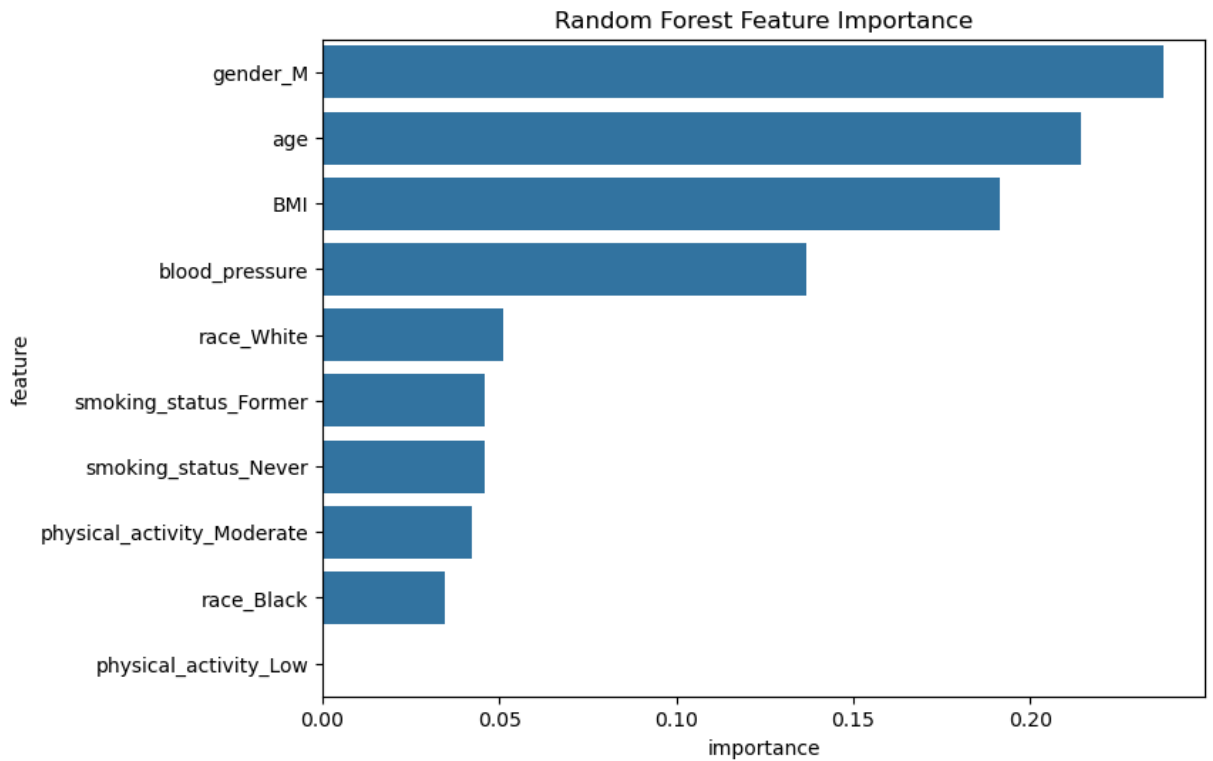
	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	1.00	1.00	1.00	1
accuracy			1.00	2
macro avg	1.00	1.00	1.00	2
weighted avg	1.00	1.00	1.00	2

```
In [19]: # =====
# 10. Feature Importance
# =====
# Random Forest feature importance
feature_importances = pd.DataFrame({
    'feature': X_train.columns,
    'importance': rf_model.feature_importances_
}).sort_values(by='importance', ascending=False)

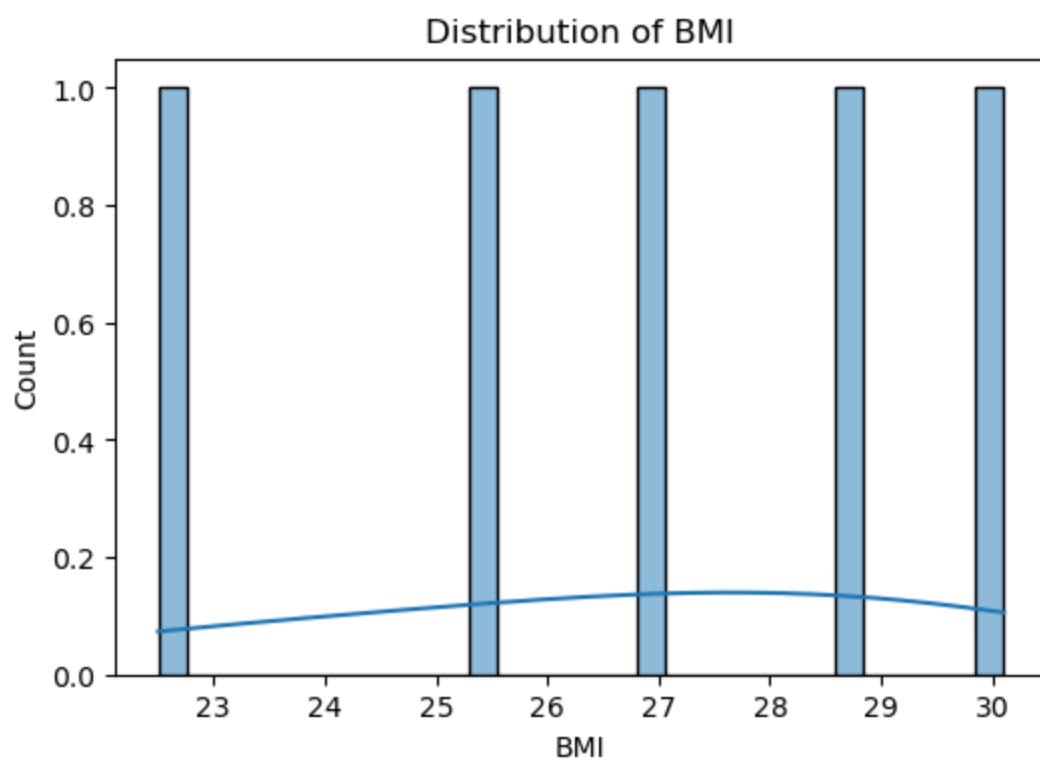
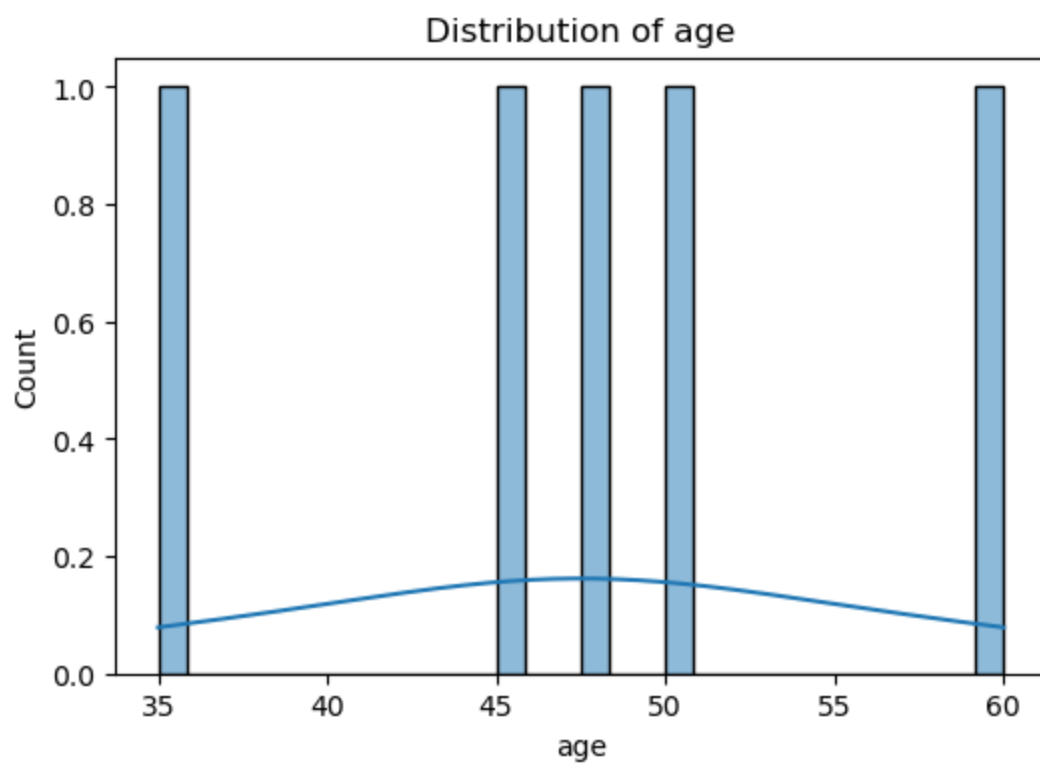
plt.figure(figsize=(8,6))
sns.barplot(x='importance', y='feature', data=feature_importances)
plt.title('Random Forest Feature Importance')
plt.show()

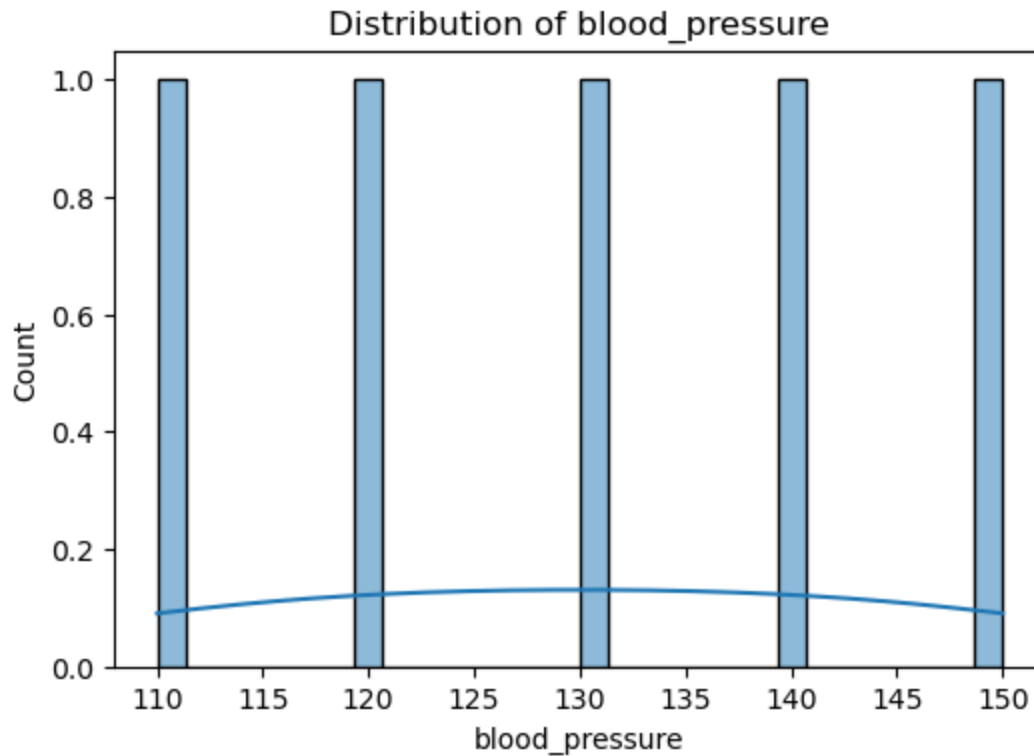
# Logistic Regression coefficients
coefficients = pd.DataFrame({
    'feature': X_train.columns,
    'coefficient': log_model.coef_[0]
}).sort_values(by='coefficient', ascending=False)

plt.figure(figsize=(8,6))
sns.barplot(x='coefficient', y='feature', data=coefficients)
plt.title('Logistic Regression Feature Coefficients')
plt.show()
```

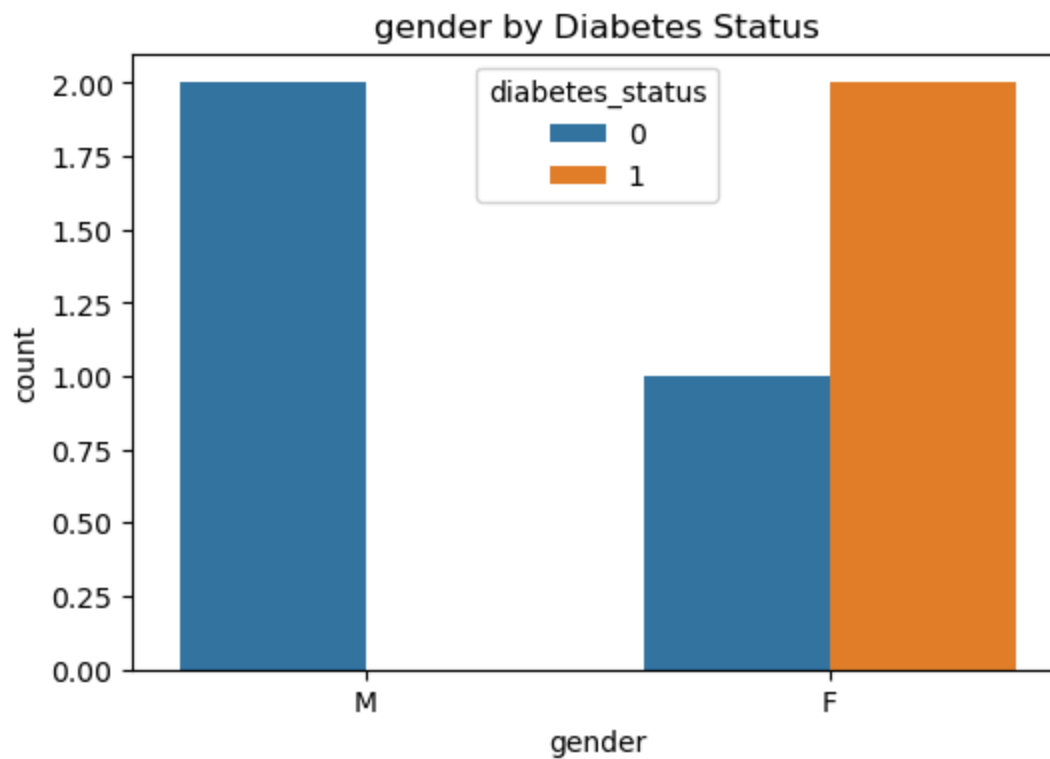


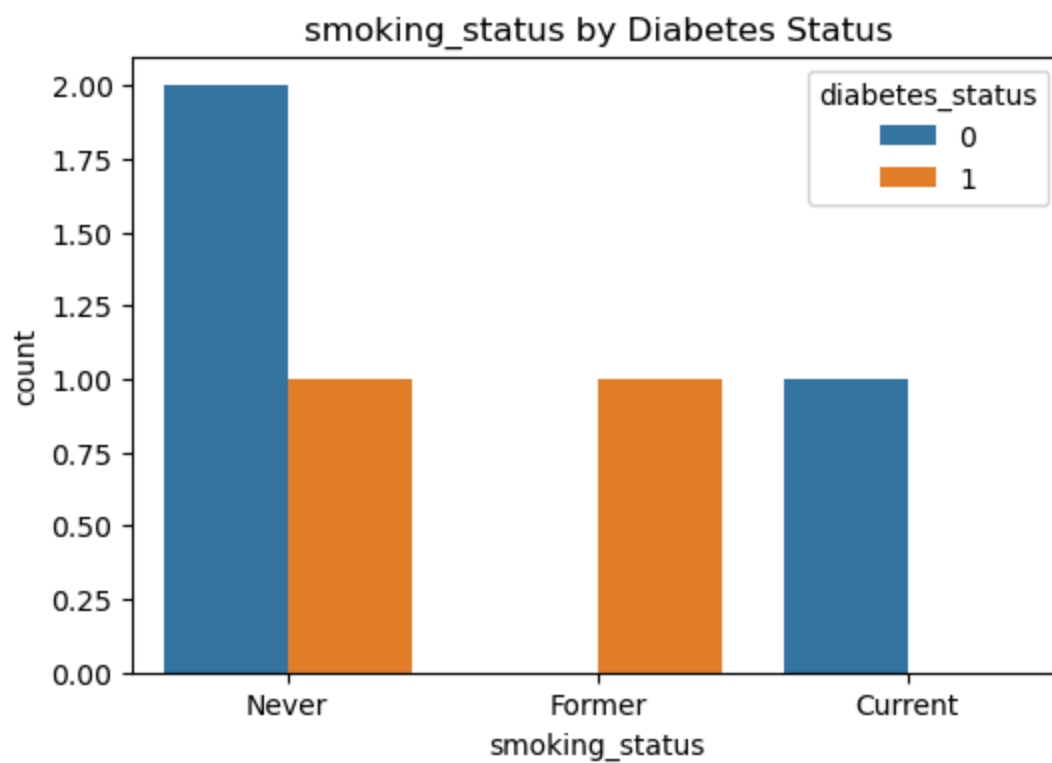
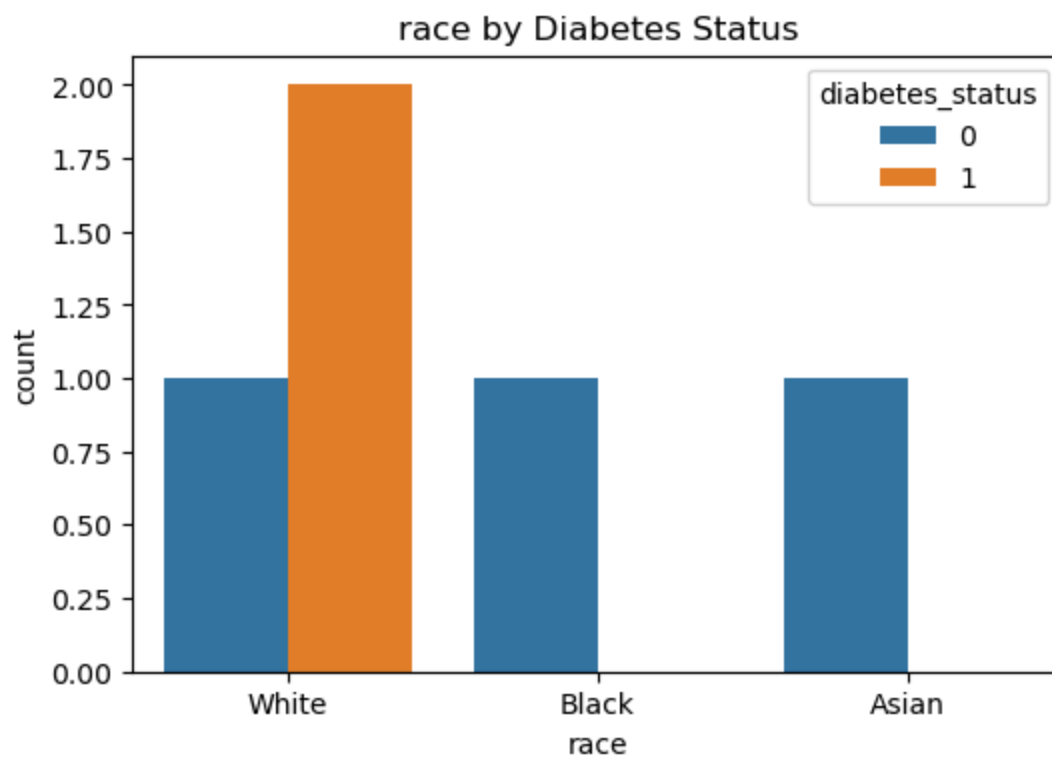
```
In [20]: # =====
# 11. Exploratory Visualizations
# =====
# Histograms for numeric features
for feature in numeric_features:
    plt.figure(figsize=(6,4))
    sns.histplot(df[feature], kde=True, bins=30)
    plt.title(f'Distribution of {feature}')
    plt.show()
```

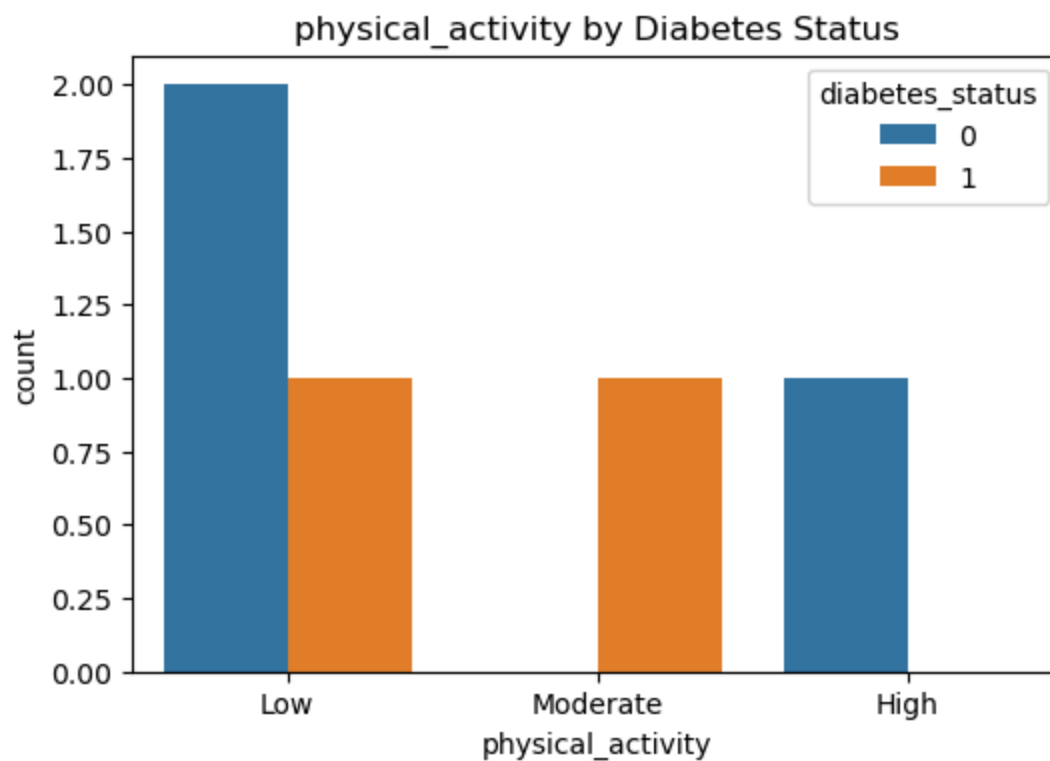




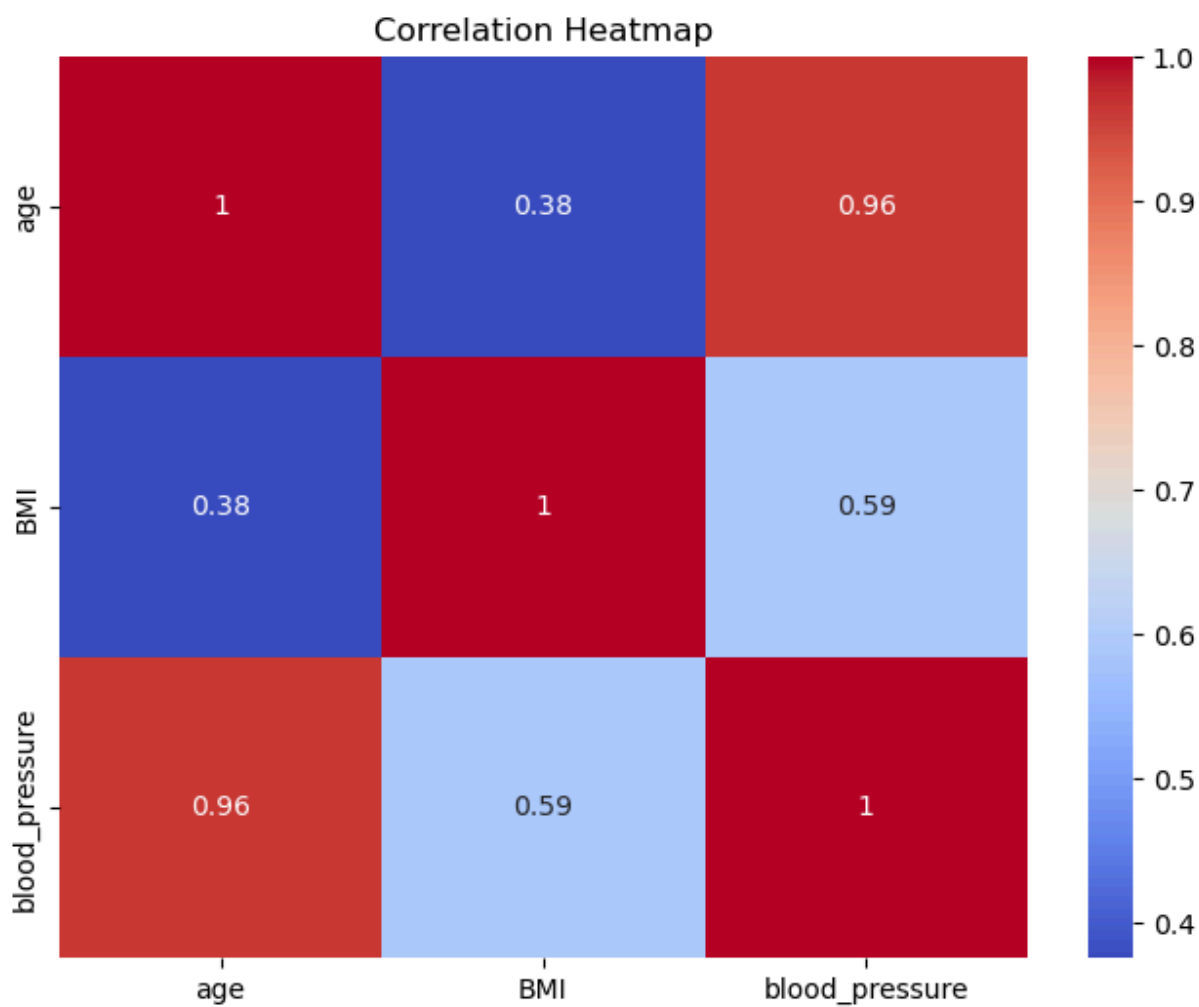
```
In [21]: # Bar plots for categorical features
for feature in categorical_features:
    plt.figure(figsize=(6,4))
    sns.countplot(x=feature, hue='diabetes_status', data=df)
    plt.title(f'{feature} by Diabetes Status')
    plt.show()
```







```
In [22]: # Correlation heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df[numeric_features].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [ ]: