UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Relevance and Anonymization for a Diabetic Dataset

PRIVACY PRESERVING INFORMATION ACCESS: HOMEWORK 1

Ferrari Luca (id. 2166294)
Scalco Riccardo (id. 2155352)

Years: 2025/2026

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

THE CONTEXT: Dataset Introduction

Datas regard Indian Women with at least 21 years old.
Those specific real cases aimed to study pattern in *epidiomologgy* , bias in the models and fairness.
Several constraints were placed on the selection of these instances from a larger database.

- **Pregnancies**: Number of times pregnant
- **Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- **BloodPressure**: Diastolic blood pressure (mm Hg)
- **SkinThickness**: Triceps skin fold thickness (mm)
- **Insulin**: 2-Hour serum insulin (mu U/ml)
- **BMI**: Body mass index (weight in kg/(height in m)^2)
- **DiabetesPedigreeFunction**: Diabetes pedigree function
- **Age**: Age (years)
- **Outcome**: Class variable (0 or 1)

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

THE CONTEXT: Goal

- This project analyzes how *privacy-preserving techniques* affect the **accuracy** of predictive models.

- We use the dataset as a benchmark for implementing machine learning algorithms as Random Forest Classifier, where the outcome is either 0 or 1.

- We examine the microdata to understand each attribute in the records, allowing us to apply proper anonymization methods to protect sensitive medical information.

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

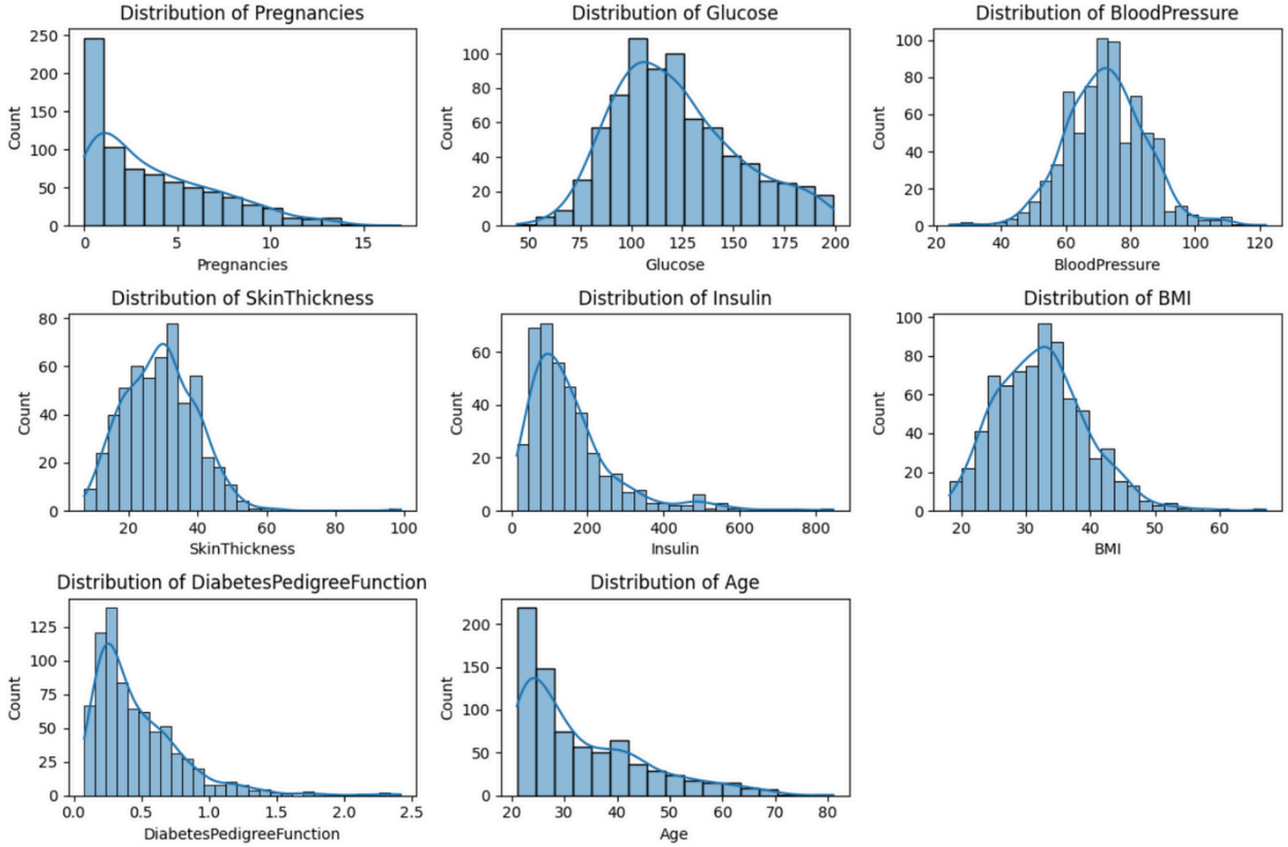
THE DATA: Type of Attribute presents:

- **Identifiers:** None are directly present.
- **Quasi-identifiers:** All attributes can become identifying when combined.
  - *Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age.*
- **Sensitive data:** Several outliers make the data potentially re-identifiable.

Outliers Problems: For example, one woman had 18 pregnancies — an unusual case that could risk revealing her identity in a small dataset.

UNIVERSITÀ DEGLI STUDI DI PADOVA

Data Distribuition



Pre-elaboration

Low-frequency combinations were likely causing noise/overfitting.

Removing them makes the training data more representative and cleaner.

UNIVERSITÀ DEGLI STUDI DI PADOVA

CODE: Macrodata techniques

```
age_pregnancies_df = diabetes_df[['Age', 'Pregnancies']].groupby('Age').value_counts()\
                        .reset_index()\
                        .pivot_table(index='Age', columns='Pregnancies', values='count')
```

SUPPRESSION

```
threshold = 2

privatized_age_pregnancies_df = age_pregnancies_df.copy()
privatized_age_pregnancies_df[age_pregnancies_df<threshold] = "SUPPRESSED"
privatized_age_pregnancies_df
```

Pregnancies	0	1	2	3	4	5	6	7
Age								
21	21.0	23.0	14.0	3.0	2.0	NaN	NaN	NaN
22	17.0	22.0	18.0	10.0	4.0	NaN	NaN	NaN
23	8.0	13.0	10.0	3.0	3.0	NaN	SUPPRESSED	NaN
24	9.0	14.0	9.0	7.0	4.0	SUPPRESSED	SUPPRESSED	SUPPRE
25	13.0	7.0	14.0	9.0	2.0	3.0	NaN	NaN
26	9.0	7.0	5.0	5.0	4.0	SUPPRESSED	2.0	NaN
27	5.0	4.0	8.0	6.0	3.0	4.0	2.0	NaN
28	2.0	8.0	5.0	6.0	5.0	4.0	5.0	NaN
29	3.0	6.0	5.0	2.0	5.0	2.0	3.0	SUPPRE
30	SUPPRESSED	3.0	2.0	4.0	4.0	5.0	SUPPRESSED	NaN
31	4.0	2.0	2.0	2.0	6.0	NaN	3.0	3.0
32	2.0	2.0	NaN	2.0	SUPPRESSED	2.0	3.0	2.0
33	2.0	4.0	SUPPRESSED	NaN	3.0	2.0	SUPPRESSED	NaN
34	NaN	NaN	SUPPRESSED	2.0	3.0	SUPPRESSED	SUPPRESSED	2.0
35	2.0	NaN	NaN	SUPPRESSED	SUPPRESSED	3.0	SUPPRESSED	NaN
36	SUPPRESSED	2.0	NaN	2.0	2.0	SUPPRESSED	SUPPRESSED	3.0

UNIVERSITÀ DEGLI STUDI DI PADOVA

N-K RULE

```
n = 3
k = 0.3
```

```
col = 'Pregnancies'

nk_df = diabetes_df[col].head(5)
print(nk_df)

contribution = nk_df / np.sum(nk_df)
print("Percentual contribution:", contribution.values)

contribution_higher_then_k = 0
for i in contribution:
    if i > k:
        contribution_higher_then_k += 1
print(contribution_higher_then_k)

is_sensitive = False
if contribution_higher_then_k > 0 and contribution_higher_then_k < n:
    is_sensitive = True
print(f"Is the cell sensitive according to ({n}, {int(k*100)}%)? {is_sensitive}")
```

```
0    6
1    1
2    8
3    1
4    0
Name: Pregnancies, dtype: int64
Percentual contribution: [0.375  0.0625 0.5    0.0625 0.    ]
2
Is the cell sensitive according to (3, 30%)? True
```

These rules help identify risky data points that could reveal individuals and guide where to apply anonymization

P-PERCENTAGE

```
p = list(range(0, 101, 1))
col = 'Pregnancies'

p_percent_df = diabetes_df[col].head(5)
print(p_percent_df)
v1, v2 = p_percent_df.nlargest(2).values
t = p_percent_df.sum()

for perc in p:
    if (t - v1 - v2) < (perc/100) * v1:
        print(f"Min p = {perc}%")
        break
```

```
0    6
1    1
2    8
3    1
4    0
Name: Pregnancies, dtype: int64
Min p = 26%
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

Predictive Model Pre Microdata

Define the Model

```
def train_model(X_train, y_train):
    rf = RandomForestClassifier(
        n_estimators=500,
        max_depth=10,
        min_samples_split=10,
        class_weight='balanced',
        random_state=42
    )
    rf.fit(X_train, y_train)
    return rf

def test_model(X_test, y_test,rf):
    y_pred_rf = rf.predict(X_test)

    print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
    print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
X = diabetes_df.drop('Outcome', axis=1)
y = diabetes_df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = train_model(X_train, y_train)
test_model(X_test, y_test, model)
```

```
Random Forest Accuracy: 0.7552083333333334
Confusion Matrix:
 [[93 30]
 [17 52]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.76      0.80       123
           1       0.63      0.75      0.69        69

    accuracy                           0.76       192
   macro avg       0.74      0.75      0.74       192
weighted avg       0.77      0.76      0.76       192
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

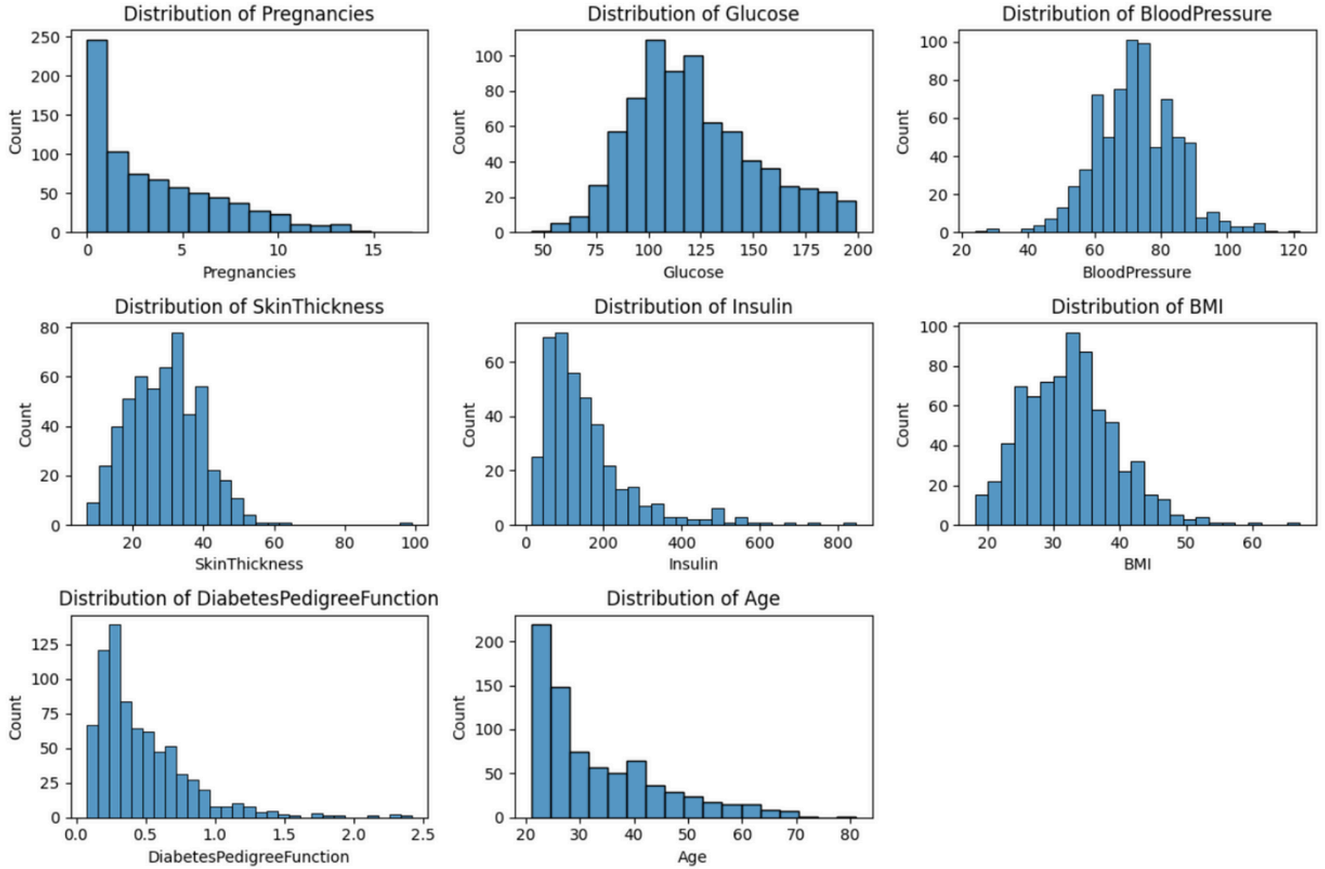
CODE: techniques Microdata Protection

Synthetic Data Injection

```
X = diabetes_micro_df.drop("Outcome", axis=1)
y = diabetes_micro_df["Outcome"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

imputer = KNNImputer(n_neighbors=5)
X_imputed = imputer.fit_transform(X_scaled)

X_restored = scaler.inverse_transform(X_imputed)
X = pd.DataFrame(X_restored, columns=X.columns)
```

```
diabetes_micro_generated_df = pd.concat([X, y], axis=1)
```

```
plt.figure(figsize=(12, 8))
for i, col in enumerate(diabetes_micro_generated_df.columns[:-1]):
    plt.subplot(3, 3, i+1)
    sns.histplot(diabetes_micro_df[col])
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```



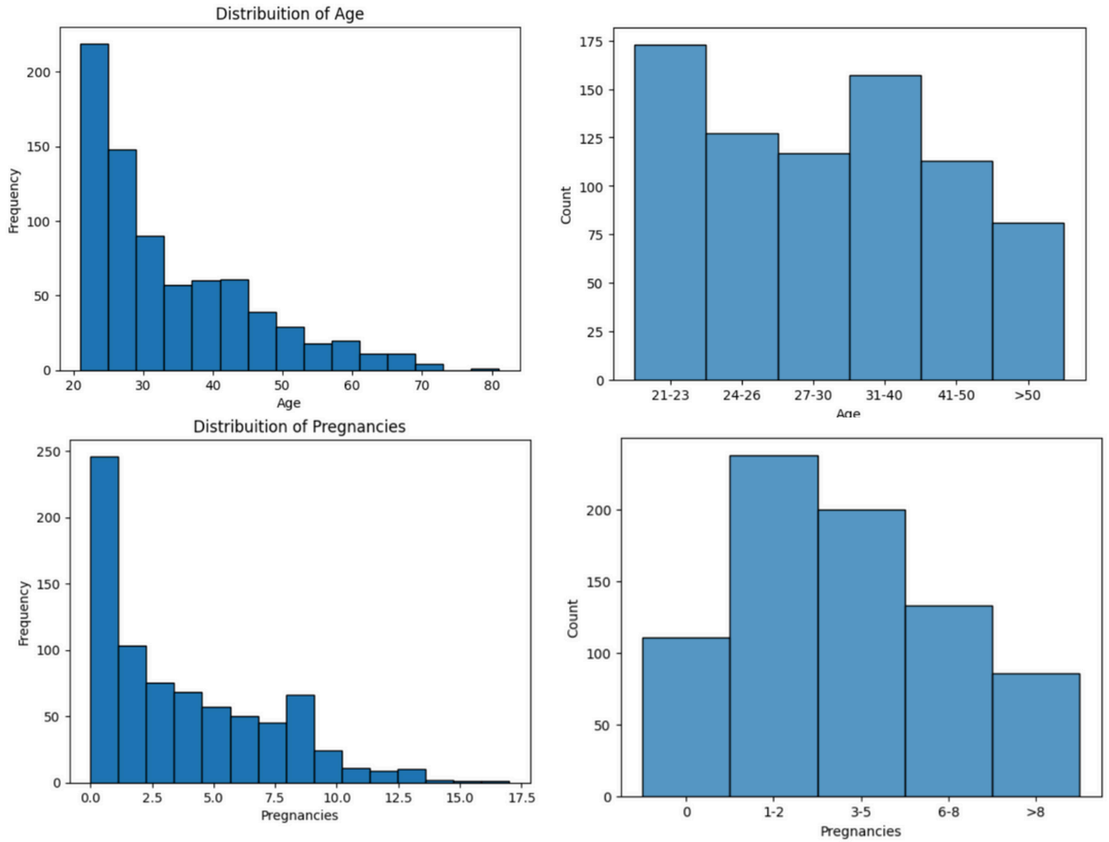
UNIVERSITÀ DEGLI STUDI DI PADOVA

CODE: techniques Microdata Protection

```python
def Generalization(bins ,labels, target, df, group_means):
    plt.hist(diabetes_micro_generated_df[target], bins=15, edgecolor='black')
    plt.title("Distribuition of "+target)
    plt.xlabel(target)
    plt.ylabel('Frequency')
    plt.show()

    df[target] = pd.cut(df[target], bins=bins, labels=labels)
    plotGroup(df, target)

    df[target] = df[target].map(group_means)

    plotGroup(df, target)

    return df
```

Generalization

```python
#Age
bins = [0, 20, 23, 26, 30, 40, 50, 100]
labels = ['<20', '21-23', '24-26', '27-30', '31-40', '41-50', '>50']
age_group_means = {
    '<20': 10,
    '21-23': 22,
    '24-26': 25,
    '27-30': 28.5,
    '31-40': 35.5,
    '41-50': 45.5,
    '>50': 60
}
diabetes_micro_df = Generalization(bins, labels, "Age", diabetes_micro_df, age_group_means)

#Pregnancies
bins = [-1, 0, 2, 5, 8, 20]
labels = ['0', '1-2', '3-5', '6-8', '>8']
pregnancies_group_means = {
    '0': 0,
    '1-2': 1.5,
    '3-5': 4,
    '6-8': 7,
    '>8': 10
}
diabetes_micro_df = Generalization(bins, labels, "Pregnancies", diabetes_micro_df, pregnancies_gr
oup_means)
```



UNIVERSITÀ DEGLI STUDI DI PADOVA

Top/Bottom Coding

```
diabetes_micro_df['Glucose'] = diabetes_micro_df['Glucose'].clip(lower=60, upper=190)

diabetes_micro_df['Insulin'] = diabetes_micro_df['Insulin'].clip(upper=300)
```

Rounding

```
diabetes_micro_df['Insulin'] = (diabetes_micro_df['Insulin'] / 10).round() * 10

diabetes_micro_df['BloodPressure'] = (diabetes_micro_df['BloodPressure'] / 5).round() * 5

diabetes_micro_df['Glucose'] = (diabetes_micro_df['Glucose'] / 5).round() * 5
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

FINAL: Results

Without Anonymization

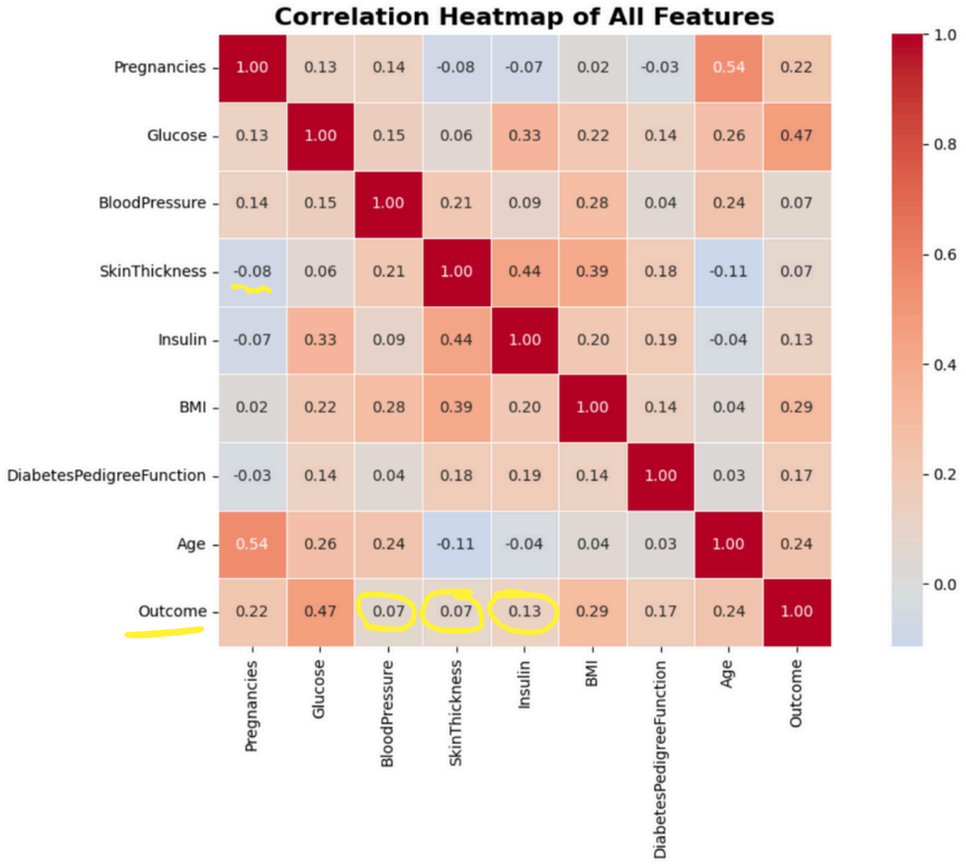
```
Random Forest Accuracy: 0.7552083333333334
Confusion Matrix:
 [[93 30]
 [17 52]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.76      0.80       123
           1       0.63      0.75      0.69        69

    accuracy                           0.76       192
   macro avg       0.74      0.75      0.74       192
weighted avg       0.77      0.76      0.76       192
```

Final With Anonymization

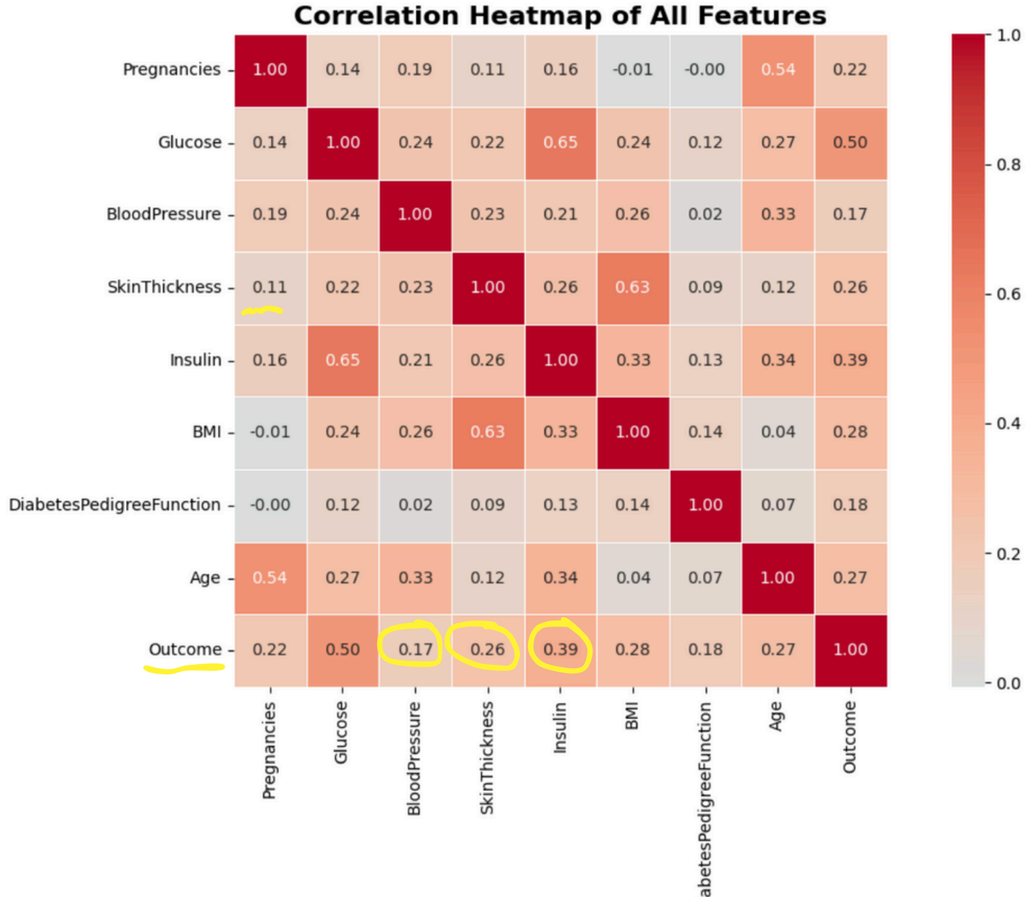
```
Random Forest Accuracy: 0.7083333333333334
Confusion Matrix:
 [[99 24]
 [32 37]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.80      0.78       123
           1       0.61      0.54      0.57        69

    accuracy                           0.71       192
   macro avg       0.68      0.67      0.67       192
weighted avg       0.70      0.71      0.70       192
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

Without Anonymization



Final With Anonymization



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Thank you for your attention