Group 15 522 Project

Inder Khera, Jenny Zhang, Jessica Kuo, Javier Martinez

Why is Diabetic Study Important?

- Diabetes is a widespread condition affecting millions globally
- Identifying predictors helps focus on preventive measures for at-risk populations
- Different populations may have unique risk factors
- Reduce healthcare costs and improve the quality of life by preventing long-term complications

---

Previous Research

- Use of Neural Networks (Jack W Smith, et. al)
- Logistic Regression (UCI Machine Learning)
- Random Forestion (Quan Zou, et. al)

What is unique about our study?

- Using Logistic Regression for easier interpretation
- Use of Data Validation and Class Imbalance for better accuracy

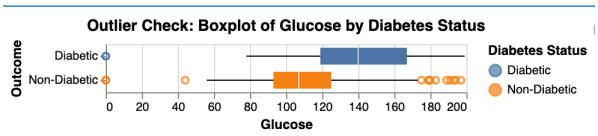
---

Why use a Logistic Regression model?

- Study about predicting whether an individual has diabetes or not
  - Binary Classification
- Provides feature coefficients that directly reflect the importance and direction
- Focis is on probabilities rather than hard classifications
  - Can help clinicians determine confidence level of a diagnosis

---

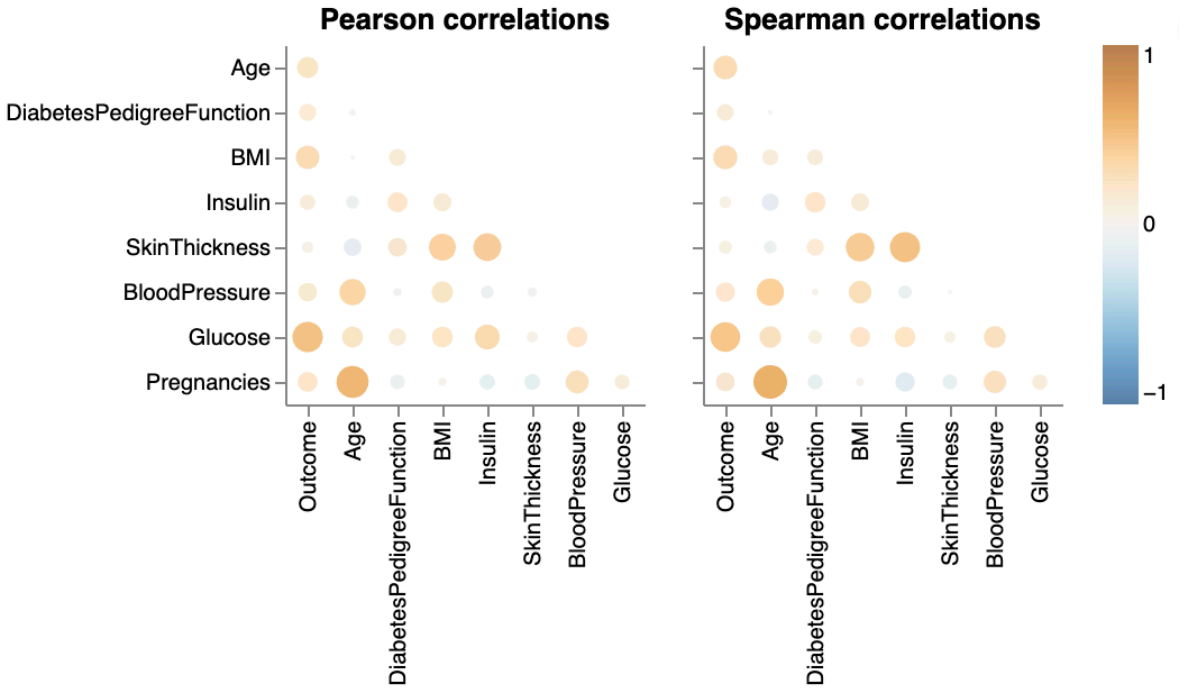
Data Validation - Pandera



- Ensures data integrity: Free of outliers & invalid values
- Based on medically plausible values:
  - Glucose & Blood Pressure can't be 0 if person is alive

```
schema = pa.DataFrameSchema(
    {
        "Outcome": pa.Column(int, pa.Check.isin([0, 1])),
        "Pregnancies": pa.Column(int, pa.Check.between(0, 15), nullable=True),
        "Glucose": pa.Column(int, pa.Check.between(50, 240), nullable=True),
        "BloodPressure": pa.Column(int, pa.Check.between(40, 180), nullable=True),
        "SkinThickness": pa.Column(int, pa.Check.between(0, 80), nullable=True),
        "Insulin": pa.Column(int, pa.Check.between(0, 800), nullable=True),
        "BMI": pa.Column(float, pa.Check.between(0, 65), nullable=True),
        "DiabetesPedigreeFunction": pa.Column(float, pa.Check.between(0, 2.5), nullable=True),
        "Age": pa.Column(int, pa.Check.between(18, 90), nullable=True),
    }
```

Data Validation - Deepchecks




```
check_feat_lab_corr = FeatureLabelCorrelation().add_condition_feature_pps_less_than(0.7)
check_feat_lab_corr_result = check_feat_lab_corr.run(dataset = train_df_ds)

check_feat_feat_corr = FeatureFeatureCorrelation().add_condition_max_number_of_pairs_above_threshold(
    threshold = 0.7, n_pairs = 0)
check_feat_feat_corr_result = check_feat_feat_corr.run(dataset = train_df_ds)

if not check_feat_lab_corr_result.passed_conditions():
    raise ValueError("Feature-Label correlation exceeds the maximum acceptable threshold.")

if not check_feat_feat_corr_result.passed_conditions():
    raise ValueError("Feature-feature correlation exceeds the maximum acceptable threshold.")
```

- Correlations close to standard multicollinearity threshold of 0.7
- Use Deepchecks to ensure no multicollinearity is occuring