

Diabetes Prediction

```
In [1]: import numpy as np
import pandas as pd
import requests
import altair as alt
import os
from sklearn.preprocessing import StandardScaler
from sklearn.compose import make_column_transformer, make_column_selector
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.pipeline import make_pipeline
from ucimlrepo import fetch_ucirepo
from sklearn.metrics import (
    accuracy_score, make_scorer, fbeta_score,
    recall_score, precision_score, ConfusionMatrixDisplay
)
alt.data_transformers.disable_max_rows()

np.random.seed(522)
```

Summary

In this project we attempt to build a model to predict diabetes disease. We compared a decision tree model and naive bayes model and found the decision tree is stronger in this context. We used f2-score as our scoring function because detecting diabetes is the priority: a false negative could be much worse than a false positive.

In the test dataset: the decision tree model correctly detected 8283 of 10604 positive cases (recall rate is about 78%). This result does come at a fairly significant cost in terms of false positives (precision rate is about 30%) with 19650 false positives. Depending on the actual cost of false positive this may need significant improvement to be a viable screening model.

Introduction

In Canada and the USA approximately 10% of people are living with diabetes. In Canada in 2023 approximately 3.7 million people were living with diabetes and in the USA in 2021 approximately 38.4 million people were living with diabetes. In the USA it is the 8th leading cause of death. Globally an estimated 44% of people living with diabetes are undiagnosed.

In this project we try to predict diabetes disease based on common health factors. A reliable model could help to prescreen people and recommend following up with a

physician for people who are at risk. Given the large number of people living with undiagnosed diabetes this could potentially have a significant positive impact of world health.

The analysis uses the American CDC Behavioural Risk Factor Surveillance System (BRFSS) 2015 Diabetes Health Indicators dataset (UCI ID 891), containing 253,680 survey responses with 21 health-related features and a binary diabetes outcome (0 = no diabetes/pre-diabetes, 1 = diabetes).

No missing values were present and all features were already encoded numerically. The target classes is imbalanced ($\approx 86\%$ non-diabetic, $\approx 14\%$ diabetic).

Methods and Results

EDA

Group-wise mean differences revealed the strongest risk factors for diabetes:

- PhysHlth (days of poor physical health)
- BMI
- Age
- MentHlth (days of poor mental health)
- GenHlth (self-rated general health)

Weakest factors

- HvyAlcoholConsump
- Fruits
- Veggies
- PhysActivity
- Education
- Income

Box plots of the top five predictors clearly separate the diabetic and non-diabetic groups.

Modeling Approach

The data were split 70/30 into training and test sets with stratification on the target. Two classifiers were trained and tuned using 5-fold cross-validated grid search with **f2-score** as the scoring metric. We chose to use f2-score because it is more appropriate than accuracy or f1 because we don't want to miss true positives.

1. Decision Tree (class_weight='balanced')

Hyperparameters: max_depth $\in \{6, 8, 10, 12, 14\}$, min_samples_leaf $\in \{175, 200, 225\}$,

```

250}

Best parameters: max_depth=10, min_samples_leaf=200
Best CV f2-score = 0.5908

2. Bernoulli Naive Bayes (with StandardScaler preprocessing)
Hyperparameters: alpha ∈ {1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4}
Best parameters: alpha=1e-3
Best CV f2-score = 0.4453

```

Results

Model	Test Accuracy	Test f2-score	Test recall	Test precision
Decision Tree	0.706	0.587	0.783	0.293
Naive Bayes	0.814	0.460	0.489	0.373

Table: 1

Load Data

```
In [2]: # fetch dataset
cdc_diabetes_health_indicators = fetch_ucirepo(id=891)

# data (as pandas dataframes)
X = cdc_diabetes_health_indicators.data.features
y = cdc_diabetes_health_indicators.data.targets
```

Save Raw Data

```
In [3]: # check if raw folder exists
raw_data_path = "../data/raw"

if not os.path.exists(raw_data_path):
    os.makedirs(raw_data_path)

## Save Raw Data
X.to_csv("../data/raw/diabetes_raw_features.csv")
y.to_csv("../data/raw/diabetes_raw_targets.csv")
```

Data Wrangling

```
In [4]: # No major cleaning needed – dataset is already very clean!
# Combine features and targets to get a overview of the full data set
df = X.copy()
df['diabetes'] = y

# Quick info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253680 entries, 0 to 253679
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   HighBP          253680 non-null   int64  
 1   HighChol        253680 non-null   int64  
 2   CholCheck       253680 non-null   int64  
 3   BMI             253680 non-null   int64  
 4   Smoker          253680 non-null   int64  
 5   Stroke          253680 non-null   int64  
 6   HeartDiseaseorAttack 253680 non-null   int64  
 7   PhysActivity    253680 non-null   int64  
 8   Fruits          253680 non-null   int64  
 9   Veggies         253680 non-null   int64  
 10  HvyAlcoholConsump 253680 non-null   int64  
 11  AnyHealthcare   253680 non-null   int64  
 12  NoDocbcCost    253680 non-null   int64  
 13  GenHlth         253680 non-null   int64  
 14  MentHlth        253680 non-null   int64  
 15  PhysHlth        253680 non-null   int64  
 16  DiffWalk        253680 non-null   int64  
 17  Sex              253680 non-null   int64  
 18  Age              253680 non-null   int64  
 19  Education        253680 non-null   int64  
 20  Income           253680 non-null   int64  
 21  diabetes         253680 non-null   int64  
dtypes: int64(22)
memory usage: 42.6 MB
```

In [5]: `df.head()`

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysA
0	1	1	1	40	1	0	0	0
1	0	0	0	25	1	0	0	0
2	1	1	1	28	0	0	0	0
3	1	0	1	27	0	0	0	0
4	1	1	1	24	0	0	0	0

5 rows × 22 columns

In [6]: `df.tail()`

Out [6]:

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack
253675	1	1	1	45	0	0	0
253676	1	1	1	18	0	0	0
253677	0	0	1	28	0	0	0
253678	1	0	1	23	0	0	0
253679	1	1	1	25	0	0	1

5 rows × 22 columns

Data Summary

All the features in this dataset were selected by clinician to be relevant to a person having diabetes. We will use all features in our model.

In [7]: `df.describe()`

	HighBP	HighChol	CholCheck	BMI	Smoker
count	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000
mean	0.429001	0.424121	0.962670	28.382364	0.44316
std	0.494934	0.494210	0.189571	6.608694	0.49676
min	0.000000	0.000000	0.000000	12.000000	0.00000
25%	0.000000	0.000000	1.000000	24.000000	0.00000
50%	0.000000	0.000000	1.000000	27.000000	0.00000
75%	1.000000	1.000000	1.000000	31.000000	1.00000
max	1.000000	1.000000	1.000000	98.000000	1.00000

8 rows × 22 columns

Visualizations

In [8]: `diabetes_count = pd.DataFrame(df['diabetes'].value_counts()).reset_index()`

```
alt.Chart(diabetes_count).mark_bar().encode(
    x=alt.X('diabetes:O', title='Has Diabetes'),
    y="count",
    color="diabetes:N"
).properties(title='Count of Diabetes vs Non-Diabetes Records in Dataset')
```

Out[8]: Count of Diabetes vs Non-Diabetes Records in Dataset

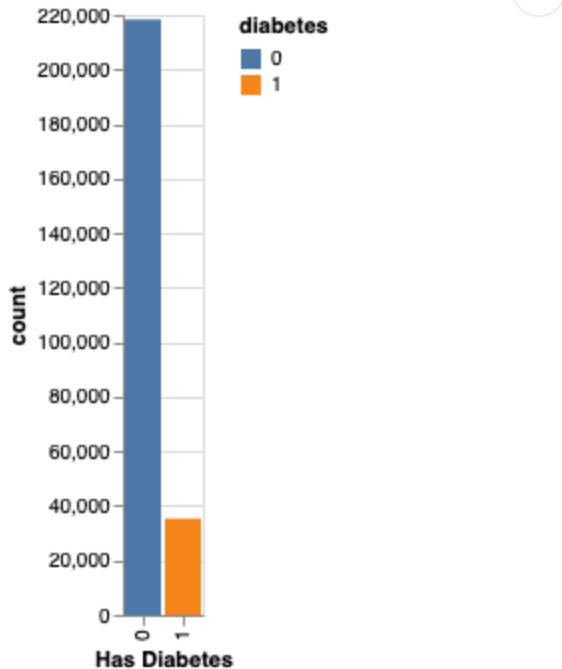


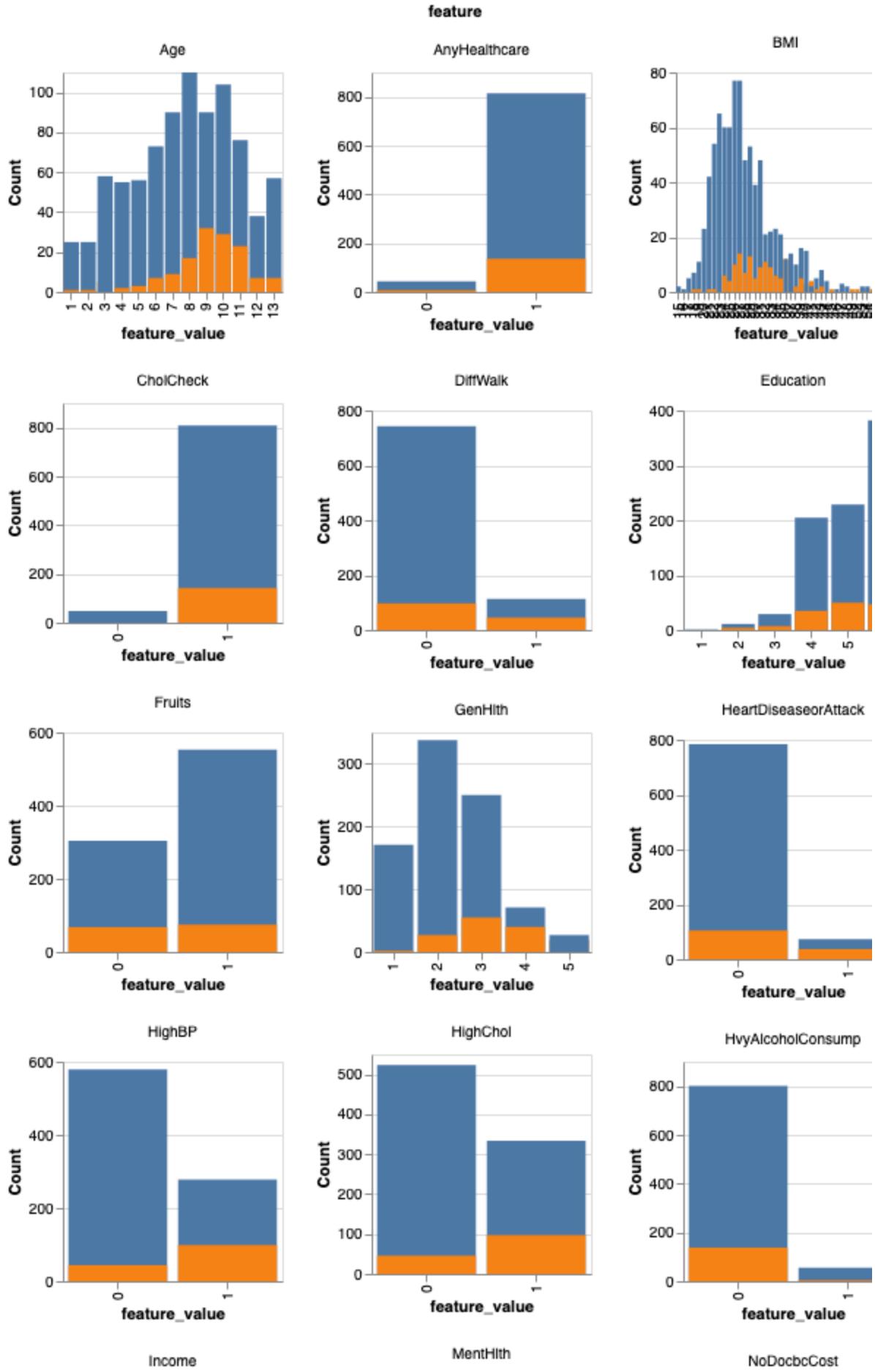
Figure 1

```
In [9]: df_sample = df.sample(n=1000, random_state=522)
features = df_sample.columns.to_list()
df_sample_long = pd.melt(df_sample, id_vars=["diabetes"], value_vars=features)

histograms = alt.Chart(df_sample_long).mark_bar().encode(
    x=alt.X("feature_value:O", # Choose to use ordinal instead of quantitative),
    y=alt.Y("count()", title="Count").stack(False),
    color=alt.Color("diabetes:N"),
).properties(
    width=150,
    height=150,
).facet(
    "feature:N",
    columns=3,
).resolve_scale(
    x="independent",
    y="independent",
).properties(
    title='Histograms of Features',
)

histograms
```

Out [9] : **Histograms of Features**



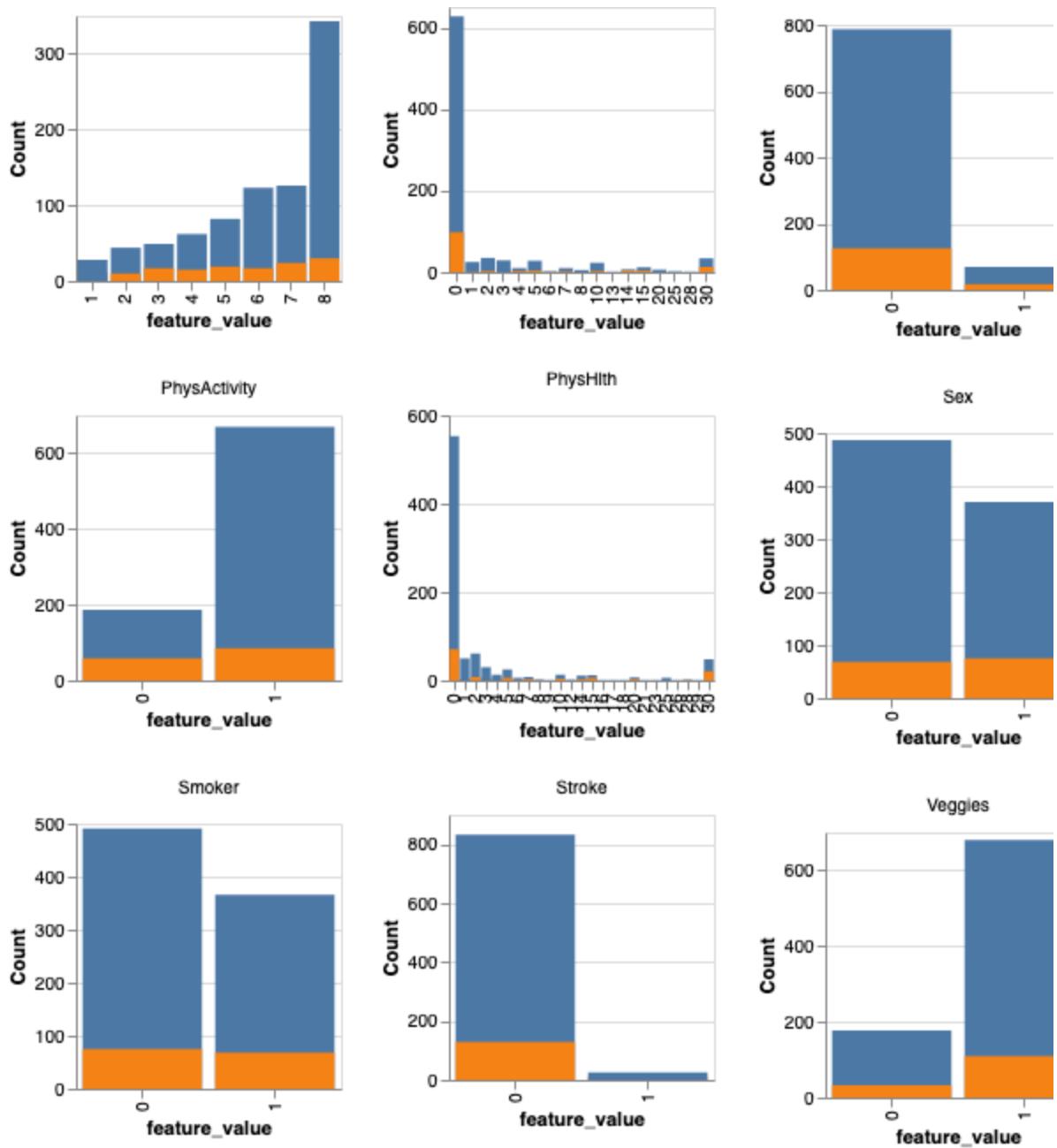


Figure 2

```
In [10]: # For binary features boxplot is not informative
non_binary_features = ['BMI', 'GenHlth', 'MentHlth', 'PhysHlth', 'Age', 'Edu'
df_sample_nonbinary = df_sample_long[df_sample_long["feature"].isin(non_bina

alt.Chart(df_sample_nonbinary).mark_boxplot().encode(
    x='diabetes:N',
    y='feature_value:Q',
    color='diabetes:N'
).facet(
    column='feature:N'
).properties(
    title='Boxplots for Non-Binary Features',
).resolve_scale(
```

```
y="independent"  
)
```

Out [10]: **Boxplots for Non-Binary Features**

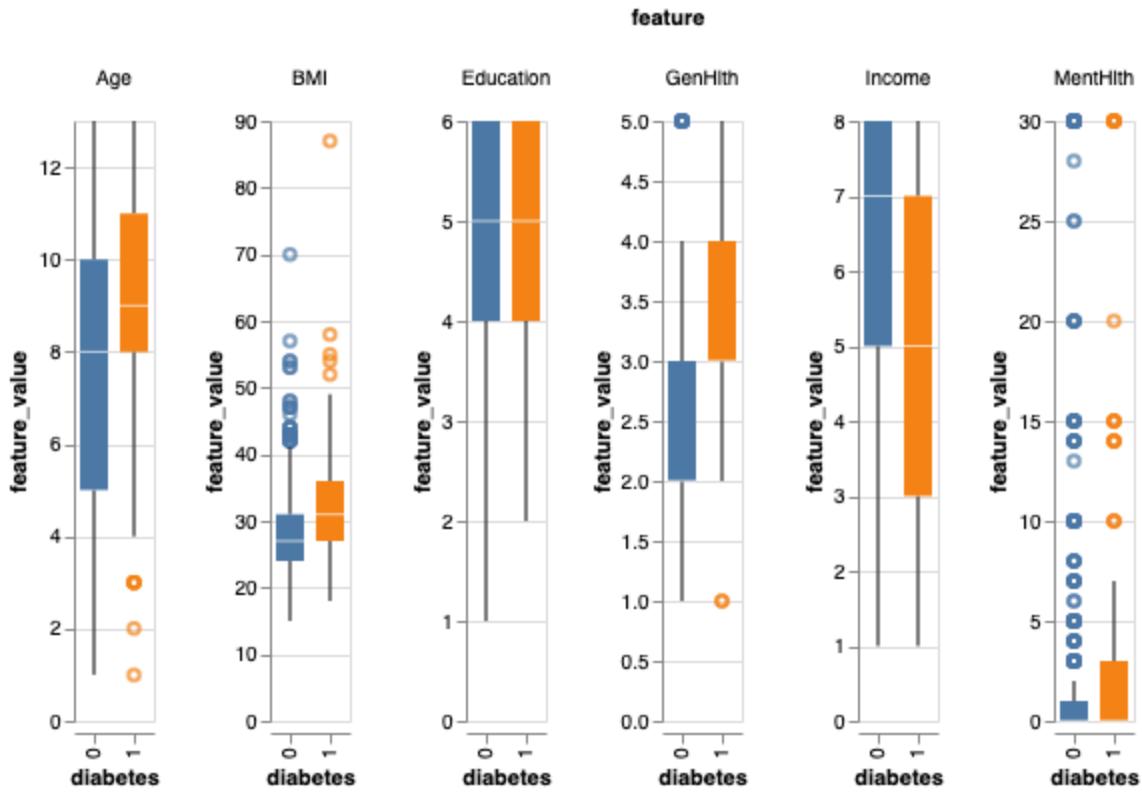


Figure 3

```
In [11]: # split the data 70-30 split  
train_df, test_df = train_test_split(  
    df, test_size=0.3, random_state=522, stratify=df['diabetes'])  
  
# check if processed folder exists  
processed_data_path = "../data/processed"  
  
if not os.path.exists(processed_data_path):  
    os.makedirs(processed_data_path)  
  
# Save processed data  
train_df.to_csv(processed_data_path+"/diabetes_train.csv", index=False)  
test_df.to_csv(processed_data_path+"/diabetes_test.csv", index=False)
```

```
In [12]: X_train = train_df.drop('diabetes', axis=1)  
y_train = train_df['diabetes']  
X_test = test_df.drop('diabetes', axis=1)  
y_test = test_df['diabetes']
```

Classification Analysis

```
In [13]: f2_scorer = make_scorer(fbeta_score, beta=2)
```

```
In [14]: tree = DecisionTreeClassifier(random_state=522, class_weight='balanced')

tree_params = {
    'max_depth': [6, 8, 10, 12, 14],
    'min_samples_leaf': [175, 200, 225, 250]
}

tree_grid = GridSearchCV(tree, tree_params, cv=5, scoring=f2_scorer, n_jobs=-1)
tree_grid.fit(X_train, y_train)

best_tree = tree_grid.best_estimator_
print("Best Decision Tree params:", tree_grid.best_params_)
print("Best CV f2-score:", tree_grid.best_score_.round(4))
```

```
Best Decision Tree params: {'max_depth': 10, 'min_samples_leaf': 225}
Best CV f2-score: 0.5919
```

```
In [15]: preprocessor = make_column_transformer(
    (StandardScaler(), X_train.columns)
)

nb_pipe = make_pipeline(
    preprocessor,
    BernoulliNB()
)

nb_params = {'bernuullinb_alpha': [1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3, 1e4]}

knn_grid = GridSearchCV(nb_pipe, nb_params, cv=5, scoring=f2_scorer, n_jobs=-1)
knn_grid.fit(X_train, y_train)

best_nb = knn_grid.best_estimator_
print("Best NB k:", knn_grid.best_params_)
print("Best CV f2-score:", knn_grid.best_score_.round(4))
```

```
Best NB k: {'bernuullinb_alpha': 0.001}
Best CV f2-score: 0.4604
```

```
In [16]: models = {
    'Decision Tree': best_tree,
    'Naive Bayes': best_nb
}

results = []

for name, model in models.items():
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]

    results.append({
        'Model': name,
        'Test Accuracy': accuracy_score(y_test, y_pred).round(3),
        'Test f2-score': fbeta_score(y_test, y_pred, beta=2).round(3),
        'Test recall': recall_score(y_test, y_pred).round(3),
        'Test precision': precision_score(y_test, y_pred).round(3),
    })
```

```
score_df = pd.DataFrame(results)
score_df
```

Out[16]:

	Model	Test Accuracy	Test f2-score	Test recall	Test precision
0	Decision Tree	0.706	0.587	0.784	0.293
1	Naive Bayes	0.814	0.460	0.489	0.373

Result Visualizations

In [17]:

```
score_melt = score_df.melt(id_vars='Model', var_name='Metric', value_name='Score')

alt.Chart(score_melt).mark_bar().encode(
    x='Model:N',
    y='Score:Q',
    color='Model:N',
    column='Metric:N'
).properties(
    title='Decision Tree vs Naive Bayes Performance on Test Set'
)
```

Out[17]:

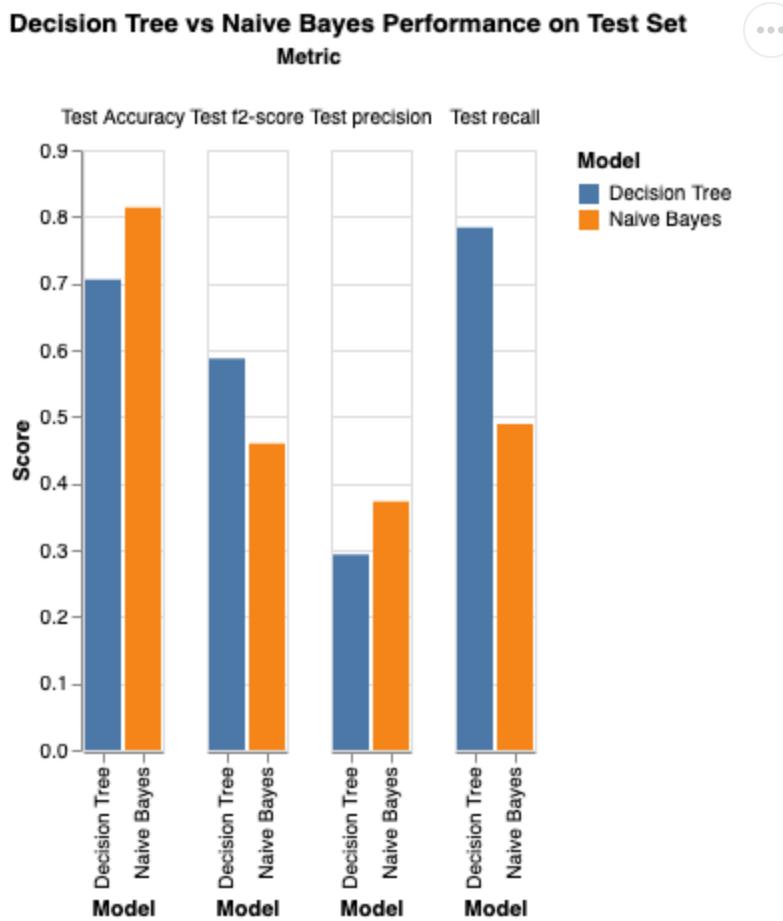


Figure 4

```
In [18]: # Confusion matrix for best model (decision tree)
```

```
ConfusionMatrixDisplay.from_estimator(  
    best_tree,  
    X_test,  
    y_test,  
    values_format="d",  
)
```

```
Out[18]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x156ab3c  
50>
```

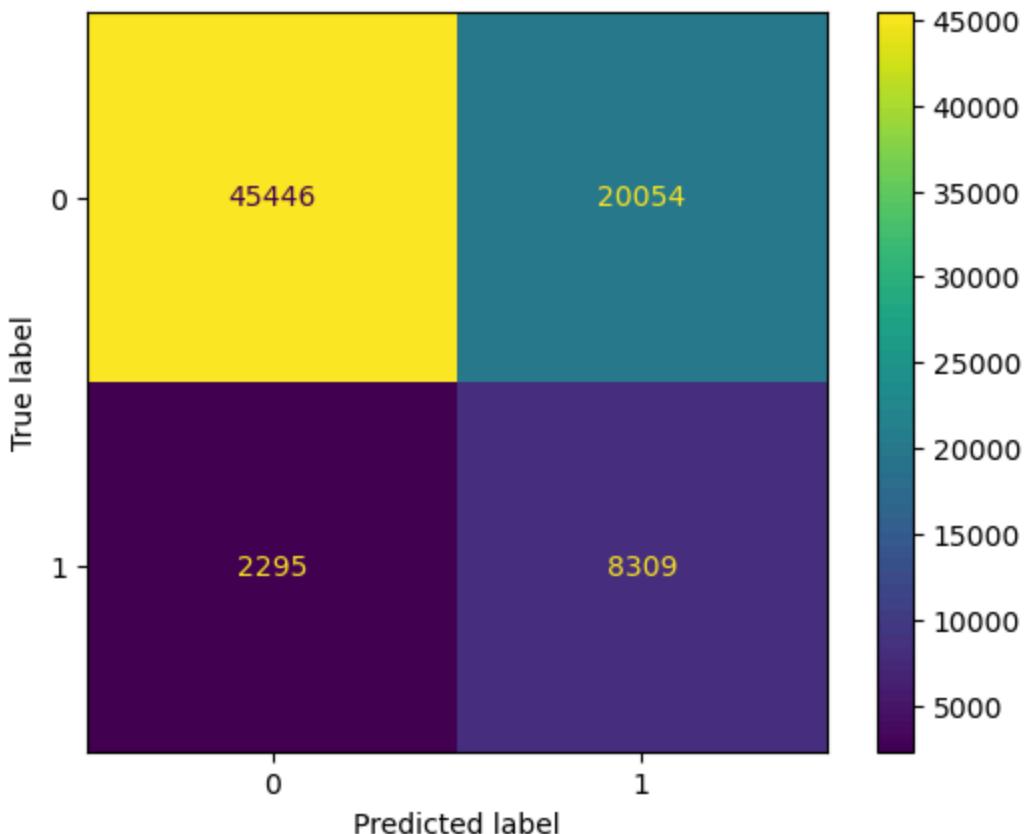


Figure 5

Discussion

The current performance of the tree model is likely already good enough to offer some benefit in the real world given the large number of people with undiagnosed diabetes. However the recall score likely could be improved on and the precision score definitely leaves something to be desired.

We were surprised by the high rate of false positives. This might be an indication of how many non-diabetic people are at risk.

Further improvements to predicting diabetes could likely be found by 1) trying a wider variety of model type and using a wider hyperparameter search 2) possibly through

more feature engineering.

A future study could be done to find a smaller set of the most easy to obtain features. Such a model would be more usable by the average person. Some work is needed to determine this smaller number of easy to obtain features that doesn't significantly reduce model performance.

Another question is if a regression model could be made that predicts a persons risk as a percent chance of developing diabetes. Longitudinal data might be required for this type of prediction.

References