Data Analysis for Diabetes Prediction Using Health Indicators

Machine Learning

Introduction

- ➔ The dataset contains detailed health indicators collected from a large population and is designed to support the analysis and prediction of diabetes risk.
- ➔ The main variable of interest is Diabetes binary, which indicates whether an individual has been diagnosed with diabetes (1) or not (0).

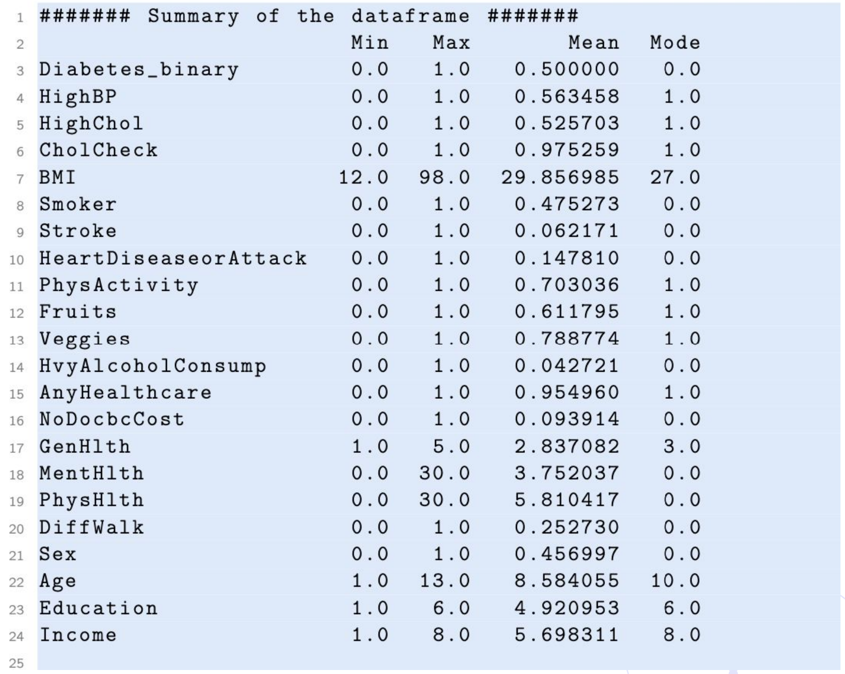
- ➔ Motivation:
  - ◆ 1. How can machine learning models be used to predict diabetes risk more accurately?
  - ◆ Can a subset of factors be used to accurately predict whether an individual has diabetes?

Selection of algorithms for the dataset

- **MLP (Multilayer Perceptron)**
  - Neural network capable of complex non-linear relationships
  - Requires tuning of hidden layers and neurons
  - Handles high-dimensional data well
- **Autoencoder + Gradient Boosting Classifier**
  - Autoencoder reduces dimensionality and extracts features
  - Gradient Boosting builds strong predictors from decision trees
  - Combined approach enhances performance for complex datasets
- **Naive Bayes**
  - Probabilistic classifier based on Bayes' theorem
  - Computationally efficient and works well with limited data
  - Assumes feature independence (may be limiting)

Preprocessing - Data cleaning

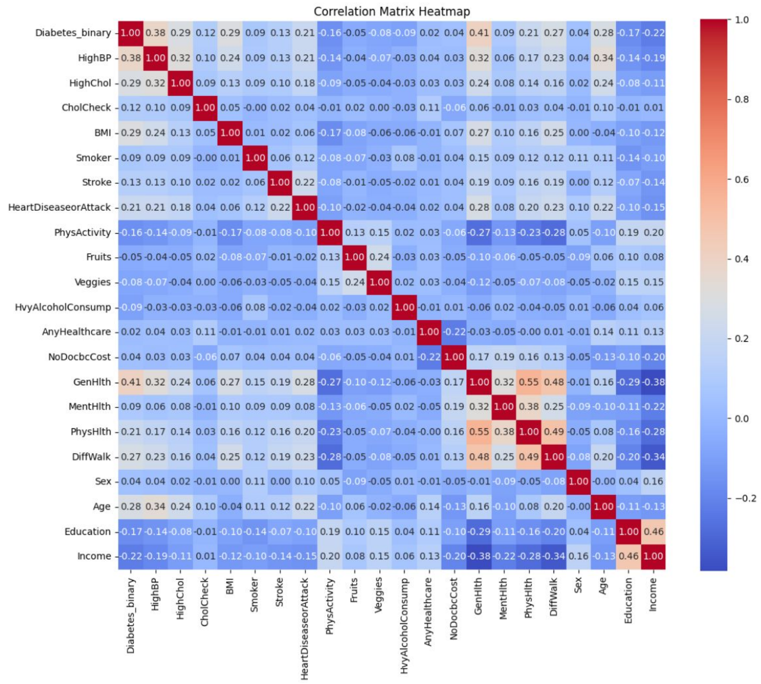
- ➔ Found 1635 **duplicate rows** , but they are not removed to ensure we preserve all relevant data for analysis.
- ➔ IQR method flagged 40,205 records as containing at least one **outlier** attribute—more than half of the dataset.
- ➔ Z-score method was far more reasonable giving 1927 outliers. We removed them, that left our dataset with 68,765 remaining records for training and testing



```
####### Summary of the dataframe #######
                       Min    Max       Mean   Mode
Diabetes_binary        0.0    1.0   0.500000    0.0
HighBP                 0.0    1.0   0.563458    1.0
HighChol               0.0    1.0   0.525703    1.0
CholCheck              0.0    1.0   0.975259    1.0
BMI                   12.0   98.0  29.856985   27.0
Smoker                 0.0    1.0   0.475273    0.0
Stroke                 0.0    1.0   0.062171    0.0
HeartDiseaseorAttack   0.0    1.0   0.147810    0.0
PhysActivity           0.0    1.0   0.703036    1.0
Fruits                 0.0    1.0   0.611795    1.0
Veggies                0.0    1.0   0.788774    1.0
HvyAlcoholConsump      0.0    1.0   0.042721    0.0
AnyHealthcare          0.0    1.0   0.954960    1.0
NoDocbcCost            0.0    1.0   0.093914    0.0
GenHlth                1.0    5.0   2.837082    3.0
MentHlth               0.0   30.0   3.752037    0.0
PhysHlth               0.0   30.0   5.810417    0.0
DiffWalk               0.0    1.0   0.252730    0.0
Sex                    0.0    1.0   0.456997    0.0
Age                    1.0   13.0   8.584055   10.0
Education              1.0    6.0   4.920953    6.0
Income                 1.0    8.0   5.698311    8.0
```

Preprocessing - Correlation

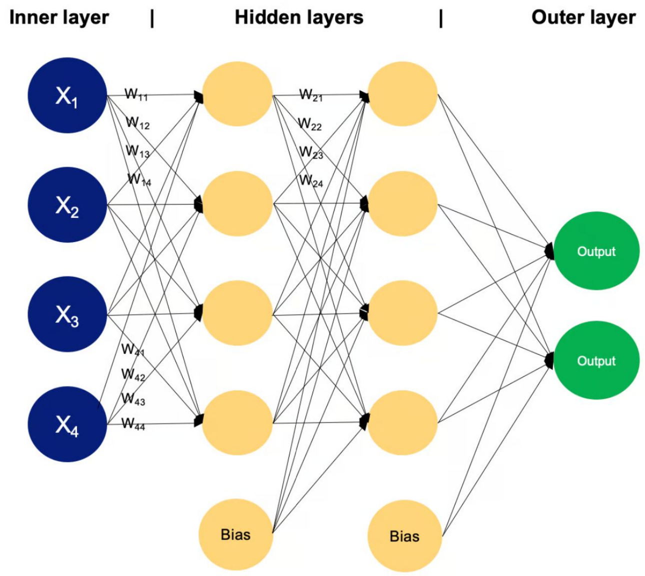
- ➔ **Data integration:** all the required data is already combined into a single CSV file.
- ➔ **Data transformation:** we apply feature scaling to the dataset in order to standardize the features.
  - ◆ StandardScaler, which standardizes the features by removing the mean and scaling to unit variance.
- ➔ **Correlation between features:**
  - ◆ Removal of weakly correlated features
  - ◆ Mitigation of multicollinearity



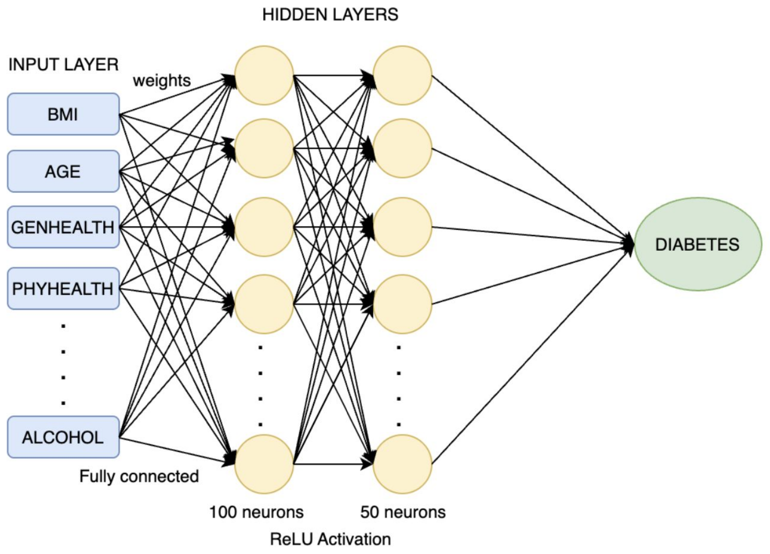
	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
Diabetes_binary	1.00	0.38	0.29	0.12	0.29	0.09	0.13	0.21	-0.16	-0.05	-0.08	-0.09	0.02	0.04	0.41	0.09	0.21	0.27	0.04	0.28	-0.17	-0.22
HighBP	0.38	1.00	0.32	0.10	0.24	0.09	0.13	0.21	-0.14	-0.04	-0.07	-0.03	0.04	0.03	0.32	0.06	0.17	0.23	0.04	0.34	-0.14	-0.19
HighChol	0.29	0.32	1.00	0.09	0.13	0.09	0.10	0.18	-0.09	-0.05	-0.04	-0.03	0.03	0.03	0.24	0.08	0.14	0.16	0.02	0.24	-0.08	-0.11
CholCheck	0.12	0.10	0.09	1.00	0.05	-0.00	0.02	0.04	-0.01	0.02	0.00	-0.03	0.11	-0.06	0.06	-0.01	0.03	0.04	-0.01	0.10	-0.01	0.01
BMI	0.29	0.24	0.13	0.05	1.00	0.01	0.02	0.06	-0.17	-0.08	-0.06	-0.06	-0.01	0.07	0.27	0.10	0.16	0.25	0.00	-0.04	-0.10	-0.12
Smoker	0.09	0.09	0.09	-0.00	0.01	1.00	0.06	0.12	-0.08	-0.07	-0.03	0.08	-0.01	0.04	0.15	0.09	0.12	0.12	0.11	0.11	-0.14	-0.10
Stroke	0.13	0.13	0.10	0.02	0.02	0.06	1.00	0.22	-0.08	-0.01	-0.05	-0.02	0.01	0.04	0.19	0.09	0.16	0.19	0.00	0.12	-0.07	-0.14
HeartDiseaseorAttack	0.21	0.21	0.18	0.04	0.06	0.12	0.22	1.00	-0.10	-0.02	-0.04	-0.04	0.02	0.04	0.28	0.08	0.20	0.23	0.10	0.22	-0.10	-0.15
PhysActivity	-0.16	-0.14	-0.09	-0.01	-0.17	-0.08	-0.08	-0.10	1.00	0.13	0.15	0.02	0.03	-0.06	-0.27	-0.13	-0.23	-0.28	0.05	-0.10	0.19	0.20
Fruits	-0.05	-0.04	-0.05	0.02	-0.08	-0.07	-0.01	-0.02	0.13	1.00	0.24	-0.03	0.03	-0.05	-0.10	-0.06	-0.05	-0.05	-0.09	0.06	0.10	0.08
Veggies	-0.08	-0.07	-0.04	0.00	-0.06	-0.03	-0.05	-0.04	0.15	0.24	1.00	0.02	0.03	-0.04	-0.12	-0.05	-0.07	-0.08	-0.05	-0.02	0.15	0.15
HvyAlcoholConsump	-0.09	-0.03	-0.03	-0.03	-0.06	0.08	-0.02	-0.04	0.02	-0.03	0.02	1.00	-0.01	0.01	-0.06	0.02	-0.04	-0.05	0.01	-0.06	0.04	0.06
AnyHealthcare	0.02	0.04	0.03	0.11	-0.01	-0.01	0.01	0.02	0.03	0.03	0.03	-0.01	1.00	-0.22	-0.03	-0.05	-0.00	0.01	-0.01	0.14	0.11	0.13
NoDocbcCost	0.04	0.03	0.03	-0.06	0.07	0.04	0.04	0.04	-0.06	-0.05	-0.04	0.01	-0.22	1.00	0.17	0.19	0.16	0.13	-0.05	-0.13	-0.10	-0.20
GenHlth	0.41	0.32	0.24	0.06	0.27	0.15	0.19	0.28	-0.27	-0.10	-0.12	-0.06	-0.03	0.17	1.00	0.32	0.55	0.48	-0.01	0.16	-0.29	-0.38
MentHlth	0.09	0.06	0.08	-0.01	0.10	0.09	0.09	0.08	-0.13	-0.06	-0.05	0.02	-0.05	0.19	0.32	1.00	0.38	0.25	-0.09	-0.10	-0.11	-0.22
PhysHlth	0.21	0.17	0.14	0.03	0.16	0.12	0.16	0.20	-0.23	-0.05	-0.07	-0.04	-0.00	0.16	0.55	0.38	1.00	0.49	-0.05	0.08	-0.16	-0.28
DiffWalk	0.27	0.23	0.16	0.04	0.25	0.12	0.19	0.23	-0.28	-0.05	-0.08	-0.05	0.01	0.13	0.48	0.25	0.49	1.00	-0.08	0.20	-0.20	-0.34
Sex	0.04	0.04	0.02	-0.01	0.00	0.11	0.00	0.10	0.05	-0.09	-0.05	0.01	-0.01	-0.05	-0.01	-0.09	-0.05	-0.08	1.00	-0.00	0.04	0.16
Age	0.28	0.34	0.24	0.10	-0.04	0.11	0.12	0.22	-0.10	0.06	-0.02	-0.06	0.14	-0.13	0.16	-0.10	0.08	0.20	-0.00	1.00	-0.11	-0.13
Education	-0.17	-0.14	-0.08	-0.01	-0.10	-0.14	-0.07	-0.10	0.19	0.10	0.15	0.04	0.11	-0.10	-0.29	-0.11	-0.16	-0.20	0.04	-0.11	1.00	0.46
Income	-0.22	-0.19	-0.11	0.01	-0.12	-0.10	-0.14	-0.15	0.20	0.08	0.15	0.06	0.13	-0.20	-0.38	-0.22	-0.28	-0.34	0.16	-0.13	0.46	1.00

Multilayer Perceptron - Theory

- MLP consists of three primary layers: the input layer, one or more hidden layers, and the output layer.
- MLPs are particularly effective in tasks where the relationships between input features and the target variable are
- non-linear, as is the case in health-related predictions like diabetes classification.

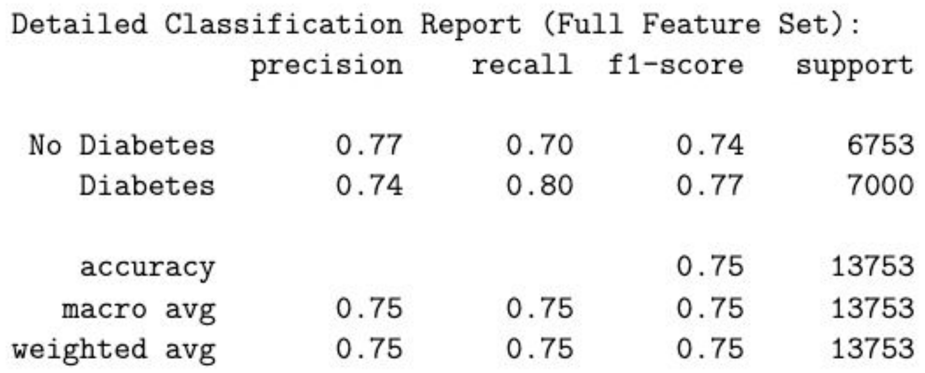


Multilayer Perceptron - Implementation

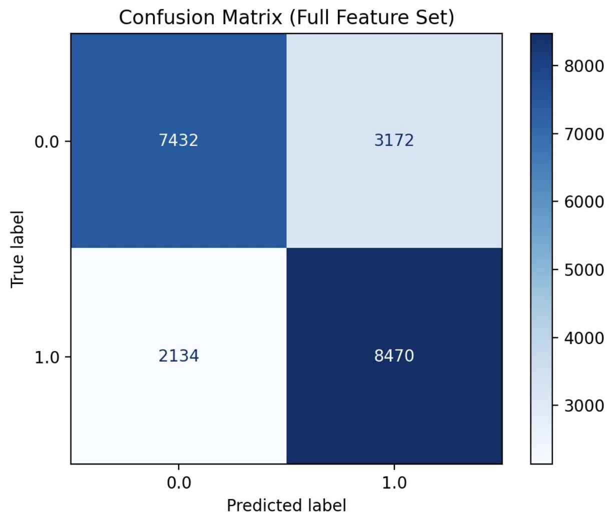


```python
# 4. Define MLP model
mlp = MLPClassifier(
    hidden_layer_sizes=(100, 50),
    activation='relu',
    solver='sgd',
    alpha=0.001,
    learning_rate='adaptive',
    max_iter=500,
    random_state=42
)

# 5. Train model
mlp.fit(X_train, y_train)

# 6. Predict on test set
y_pred = mlp.predict(X_test)

# Plot loss curve for full feature set
plt.figure(figsize=(8, 4))
plt.plot(mlp.loss_curve_)
plt.title("Loss Curve (Full Feature Set)")
plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.grid(True)
```

Multilayer Perceptron - Results

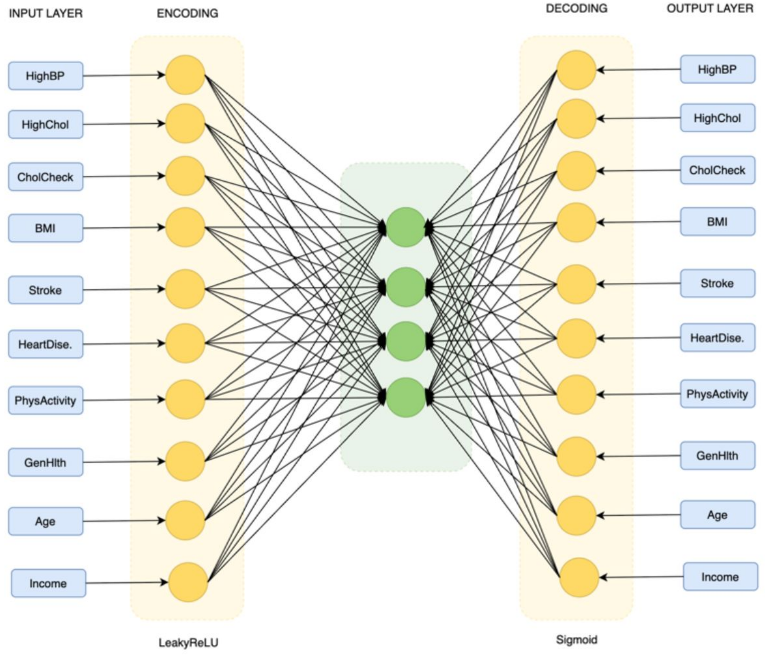


```
Detailed Classification Report (Full Feature Set):
              precision    recall  f1-score   support

 No Diabetes       0.77      0.70      0.74      6753
    Diabetes       0.74      0.80      0.77      7000

    accuracy                           0.75     13753
   macro avg       0.75      0.75      0.75     13753
weighted avg       0.75      0.75      0.75     13753
```

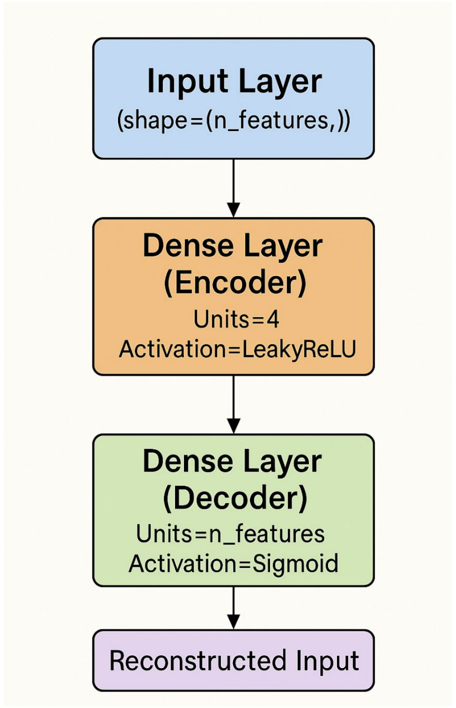


Autoencoder - Theory

➔ Autoencoders are neural networks designed to learn efficient representations of data by compressing input through an encoder, capturing essential features in a compact bottleneck layer, and reconstructing the original data using a decoder.

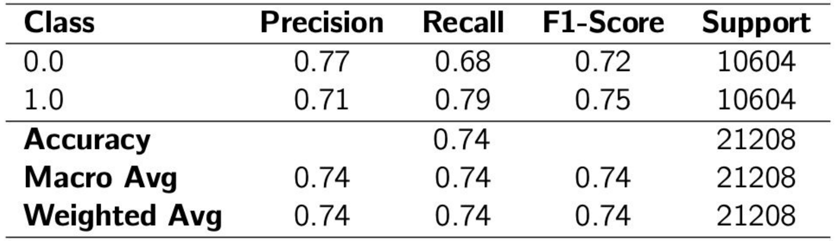


Autoencoder - Implementation

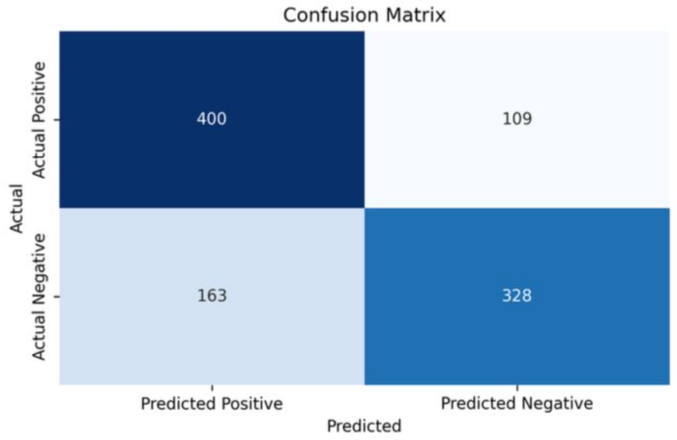


```
class Autoencoder:
    def __init__(self, encoding_dim, activation1, activation2, optimizer1, optimizer2):
        self.input_dim = X_train.shape[1]
        self.encoding_dim = encoding_dim

        input_layer = keras.layers.Input(shape=(self.input_dim,))
        encoder = keras.layers.Dense(encoding_dim, activation1)(input_layer)
        decoder = keras.layers.LeakyReLU(encoder)

        self.autoencoder = keras.Model(inputs=input_layer, outputs=decoder)
        self.encoder_model = keras.Model(inputs=input_layer, outputs=encoder)

        self.autoencoder.compile(optimizer=optimizer1, loss=optimizer2)
        self.autoencoder.summary()

    def train(self, X_train, X_test, epochs, batch_size=32):
        self.autoencoder.fit(
            X_train, X_train,
            epochs=epochs,
            batch_size=batch_size,
            shuffle=True,
            validation_data=(X_test, X_test)
        )

    def encode(self, X):
        return self.encoder_model.predict(X)
```

Autoencoder - Results



Class	Precision	Recall	F1-Score	Support
0.0	0.77	0.68	0.72	10604
1.0	0.71	0.79	0.75	10604
Accuracy		0.74		21208
Macro Avg	0.74	0.74	0.74	21208
Weighted Avg	0.74	0.74	0.74	21208



Naive Bayes - Theory

- ➔ The Naive Bayes algorithm offers a straightforward yet often surprisingly effective approach to supervised learning, particularly for classification tasks like predicting diabetes status.

- ➔ Variants:
  - Gaussian Naive Bayes
  - Multinomial Naive Bayes
  - Bernoulli Naive Bayes

$$P(\text{Class}|\text{Features}) = \frac{P(\text{Class}) \times P(\text{Features}|\text{Class})}{P(\text{Features})}$$

$$P(x_1, x_2, \ldots, x_n|\text{Class}) = P(x_1|\text{Class}) \times P(x_2|\text{Class}) \times \ldots \times P(x_n|\text{Class})$$

Naive Bayes - Implementation

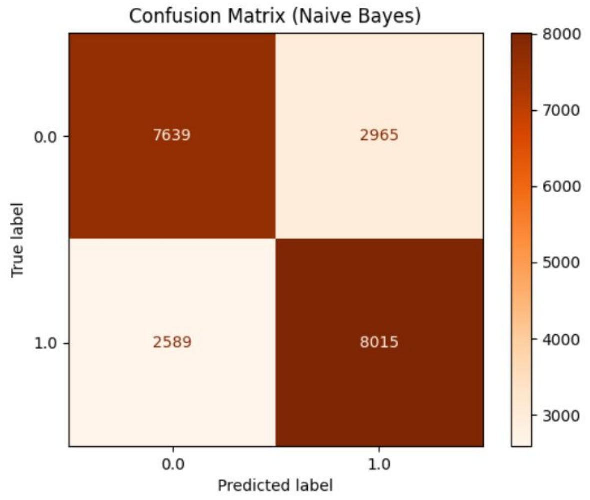
```
# Load training and test sets
X_train = pd.read_csv("datasets/X_train.csv")
X_test = pd.read_csv("datasets/X_test.csv")
y_train = pd.read_csv("datasets/y_train.csv").values.ravel()
y_test = pd.read_csv("datasets/y_test.csv").values.ravel()

# Define and train the Gaussian Naive Bayes model
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Predict on the test set
y_pred = gnb.predict(X_test)
```

```
# Evaluate performance metrics
print("Performance on Full Feature Set (Naive Bayes):")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision: {precision_score(y_test, y_pred):.4f}")
print(f"Recall: {recall_score(y_test, y_pred):.4f}")
print(f"F1 Score: {f1_score(y_test, y_pred):.4f}")

# Confusion matrix visualization
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=gnb.classes_)
disp.plot(cmap=plt.cm.Oranges)
plt.title("Confusion Matrix (Naive Bayes - Full Feature Set)")
plt.show()

# 5-Fold Cross-validation
cv_results = cross_validate(
    gnb, X_train, y_train,
    cv=5,
    scoring=["accuracy", "precision", "recall", "f1"],
    return_train_score=True
)
```

Naive Bayes - Results

Detailed Classification Report (Naive Bayes):

```
              precision    recall  f1-score   support

 No Diabetes       0.76      0.66      0.71      6753
    Diabetes       0.71      0.81      0.76      7000

    accuracy                           0.73     13753
   macro avg       0.74      0.74      0.73     13753
weighted avg       0.73      0.73      0.73     13753
```



Results & Reasoning

- ➔ We compare the three applied algorithms—Multilayer Perceptron (MLP), Autoencoder, and Gaussian Naive Bayes—using four key performance metrics: Accuracy, Precision, Recall, and F1 Score.
- ➔ In a medical setting, missing a true diabetic case (false negative) is far more critical than issuing a false alarm. Thus, we prioritize models with higher **recall**.
- ➔ Among the three models evaluated, the Multilayer Perceptron (MLP) demonstrates the best overall performance

Algorithm	**Parameters**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
MLP	hidden_layers=(100,50), activation=ReLU, solver=SGD	0.7526	0.7362	0.8011	0.7673
Autoencoder	encoder=(32,16), decoder=(16,32), activation=ReLU	0.7400	0.7100	0.7900	0.7500
Gaussian NB	default (GaussianNB)	0.7381	0.7300	0.7558	0.7427

Thank you!

Questions?

Albana Rexhepi
Eljesa Kqiku
Kaltrina Kuka