Machine Learning

What is Machine Learning?

Machine learning (ML) is a branch of artificial intelligence (AI) focused on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data.

UC Berkeley breaks out the learning system of a machine learning algorithm into three main parts.

1. *A Decision Process:* In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labeled or unlabeled, your algorithm will produce an estimate about a pattern in the data.

2. *An Error Function:* An error function evaluates the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.

3. *A Model Optimization Process:* If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this iterative "evaluate and optimize" process, updating weights autonomously until a threshold of accuracy has been met.

How does machine learning work?

Machine learning works by training algorithms on sets of data to achieve an expected outcome such as identifying a pattern or recognizing an object. Machine learning is the process of optimizing the model so that it can predict the correct response based on the training data samples.

Assuming the training data is of high quality, the more training samples the machine learning algorithm receives, the more accurate the model will become. The algorithm fits the model to the data during training, in what is called the “fitting process.” If the outcome does not fit the expected outcome, the algorithm is re-trained again and again until it outputs the accurate response. In essence, the algorithm learns from the data and reaches outcomes based on whether the input and response fit with a line, cluster, or other statistical correlation.

Types of Machine Learning

What is training data in machine learning? It depends on the type of machine learning model being used.

In broad strokes, there are three kinds of models used in machine learning.

Supervised learning :

It is a machine learning model that uses labeled training data (structured data) to map a specific feature to a label. In supervised learning, the output is known (such as recognizing a picture of an apple) and the model is trained on data of the known output. In simple terms, to train the algorithm to recognize pictures of apples, feed it pictures labeled as apples.

The most common supervised learning algorithms used today include:

- Linear regression
- Logistic regression
- K-nearest neighbors
- Naive Bayes
- Decision trees

Unsupervised learning :

It is a machine learning model that uses unlabeled data (unstructured data) to learn patterns. Unlike supervised learning, the “correctness” of the output is not known ahead of time. Rather, the algorithm learns from the data without human input (and is thus, unsupervised) and categorizes it into groups based on attributes. For instance, if the algorithm is given pictures of apples and bananas, it will work by itself to categorize which picture is an apple and which is a banana. Unsupervised learning is good at descriptive modeling and pattern matching.

The most common unsupervised learning algorithms used today include:

- K-means clustering
- Hierarchical clustering
- Dimensionality Reduction (PCA)

There’s also a mixed approach to machine learning called semi-supervised learning in which only some data is labeled. In semi-supervised learning, the algorithm must figure out how to organize and structure the data to achieve a known result. For instance, the machine learning model is told that the result is a pear, but only some training data is labeled as a pear.

Reinforcement learning :

It is a machine learning model that can be described as “learn by doing” through a series of trial and error experiments. An “agent” learns to perform a defined task through a feedback loop until its performance is within a desirable range. The agent receives positive reinforcement when it performs the task well and negative reinforcement when it performs poorly. An example of reinforcement learning is when Google researchers taught a reinforcement learning algorithm to play the game Go. The model, which had no prior knowledge of the rules of Go, simply moved pieces at random and “learned” the best moves to make. The algorithm was trained via positive and negative reinforcement to the point that the machine learning model could beat a human player at the game.

Common terminologies :

Features :

- Definition: Features (also called independent variables or predictors) are the input variables used by a machine learning algorithm to make predictions. They represent the attributes of the data that the model uses to learn.
- Example: In a dataset predicting house prices, the features could include size of the house, number of bedrooms, and location.

Labels :

- Definition: Labels (also called dependent variables or targets) are the outcomes or results that the model is trying to predict. In supervised learning, the model is trained on data that includes both features and labels.
- Example: In the house price prediction example, the label would be the price of the house.

Training Data :

- Definition: The training data is a subset of the dataset that is used to train the machine learning model. It contains both features and labels.
- Purpose: The model learns the relationship between features and labels from this data, adjusting its parameters (weights) accordingly.
- Example: If you have 1000 data points, 700 might be used for training and the remaining 300 used for testing.

Testing Data :

- Definition: The testing data is a separate subset of the dataset that is used to evaluate the model after it has been trained. It is important that the testing data is kept separate from the training data to assess how well the model generalizes to unseen data.
- Purpose: This data is used to evaluate the performance of the model and determine metrics like accuracy, precision, recall, etc.
- Example: If you split your data into 80% for training and 20% for testing, the testing data would be the 20% that is held back.

Overfitting :

- Definition: Overfitting occurs when a model learns not only the genuine patterns in the training data but also the noise or random fluctuations. The model ends up fitting the training data too well, but it performs poorly on new, unseen data.
- Cause: Overfitting usually happens when the model is too complex (e.g., too many features, too deep a decision tree).
- Signs: A high performance on the training data but a low performance on the testing data.
- Example: A decision tree that has grown too deep, capturing every small detail of the data, would overfit.

Underfitting :

- Definition: Underfitting occurs when a model is too simple to capture the underlying patterns in the data. This results in poor performance on both the training data and the testing data.
- Cause: Underfitting often occurs when the model is too simple (e.g., using a linear model for a non-linear problem) or if there is insufficient data.
- Signs: Both training and testing errors are high.
- Example: A linear regression model trying to predict a highly non-linear problem may underfit the data.

Bias :

- Definition: Bias refers to the error introduced by approximating a real-world problem with a simplified model. A high bias model makes strong assumptions about the data and might miss important patterns.
- Impact: High bias usually leads to underfitting.
- Example: A linear model used to fit a dataset that has a non-linear relationship between features and labels may have high bias.

Variance :

- Definition: Variance refers to how much the model's predictions change when trained on different subsets of the data. A model with high variance pays too much attention to the training data and may not generalize well.
- Impact: High variance usually leads to overfitting.
- Example: A decision tree with many branches might perform perfectly on the training data but fail to generalize well to unseen data due to high variance.

Model Evaluation Metrics :

These are terms that describe how well a machine learning model is performing:

- Accuracy: The percentage of correct predictions. Often used for classification tasks.
- Precision: The percentage of relevant results among the retrieved instances (true positives divided by the total predicted positives).
- Recall: The percentage of relevant results that have been retrieved (true positives divided by the total actual positives).
- F1-Score: The harmonic mean of precision and recall, often used when the class distribution is imbalanced.
- Mean Squared Error (MSE): Commonly used for regression tasks to measure the average squared difference between predicted and actual values.

Cross-Validation :

- Definition: Cross-validation is a technique used to assess the performance of a machine learning model. It involves splitting the dataset into multiple subsets (folds) and training and testing the model multiple times, each time using different training and testing data.
- Purpose: Cross-validation helps to ensure that the model performs consistently and is less prone to overfitting.
- Example: 10-fold cross-validation splits the data into 10 parts and trains the model 10 times, each time using a different 9 parts for training and the remaining 1 part for testing.

Hyperparameters :

- Definition: Hyperparameters are the parameters of the machine learning algorithm that are set before training the model (e.g., the learning rate, number of decision tree branches, or number of layers in a neural network).
- Tuning: Hyperparameter tuning involves adjusting these values to improve the performance of the model. Techniques like Grid Search or Random Search are used for hyperparameter optimization.

Feature Engineering :

- Definition: Feature engineering is the process of selecting, modifying, or creating new features from raw data to improve the model's performance.
- Example: Converting a date feature into "day of the week", "month", or "year" might help improve predictions.

Feature Selection :

- Definition: Feature selection is the process of selecting a subset of relevant features for model training, which helps to reduce dimensionality and improve the model's efficiency and performance.
- Example: Dropping irrelevant features (e.g., the "height" feature for predicting house price if height has no correlation with price).

Generalization :

- Definition: Generalization refers to the model's ability to perform well on new, unseen data. The goal of training a model is not just to perform well on the training data but also to generalize to new data.

Supervised Learning

Linear Regression

Overview :

Linear Regression is a statistical method used to establish a relationship between a dependent variable ("output" or "response") and one or more independent variables ("inputs" or "predictors"). The goal is to model this relationship as a linear equation.

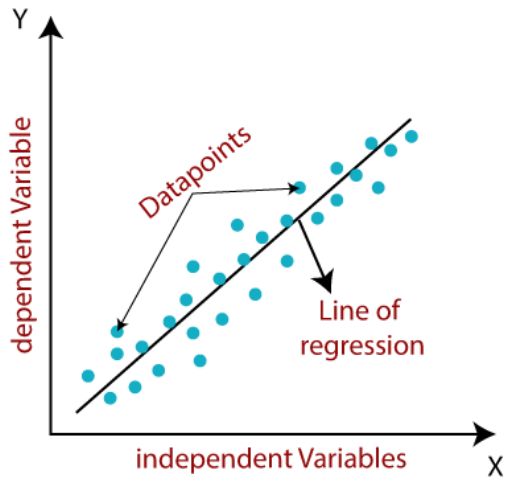
In the case of Simple Linear Regression, there is one independent variable, while Multiple Linear Regression involves two or more independent variables.

Use Cases

- Finance: Predicting stock prices or sales based on historical data.
- Healthcare: Estimating disease progression based on patient metrics.
- Marketing: Determining the impact of advertising spend on sales.
- Education: Predicting student performance based on study hours.

How it predicts continuous values

Linear regression predicts continuous values by fitting a straight line through data points, essentially assuming a linear relationship between the independent variable (input) and the dependent variable (output), allowing it to estimate a continuous value for the dependent variable based on the known value of the independent variable; this prediction is done by calculating a mathematical equation that represents this line, where the slope and intercept of the line determine the predicted value.



```
X_mean = X.mean()
print('X Mean :', X_mean)

y_mean = y.mean()
print('y Mean :', y_mean)
```

```
X Mean : 5.413333333333332
y Mean : 76004.0
```

```
numerator = np.sum((X - X_mean) * (y - y_mean))
denominator = np.sum((X - X_mean) ** 2)
slope = numerator / denominator
print('Slope     :',slope)
intercept = y_mean - slope * X_mean
print('Intercept :', intercept)
```

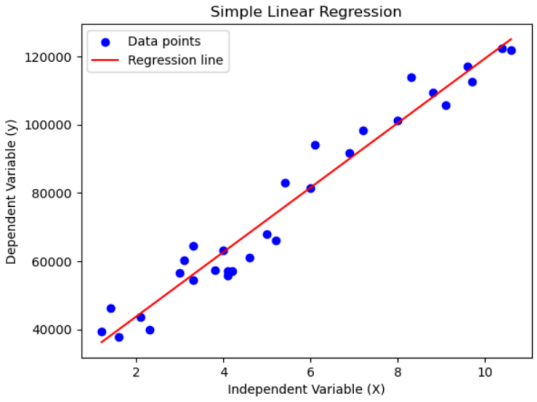
```
Slope     : 9449.962321455076
Intercept : 24848.2039665232
```

```
y_pred = slope * X + intercept
```

```
slope * 1.6 + intercept
```

```
39968.143680851324
```

```
plt.scatter(X, y, color='blue', label='Data points')
plt.plot(X, y_pred, color='red', label='Regression line')
plt.xlabel('Independent Variable (X)')
plt.ylabel('Dependent Variable (y)')
plt.title('Simple Linear Regression')
plt.legend()
plt.show()
```




```
X = data_reg[['YearsExperience']]
y = data_reg['Salary']

model_reg = LinearRegression()
model_reg.fit(X, y)
```

```
LinearRegression
LinearRegression()
```

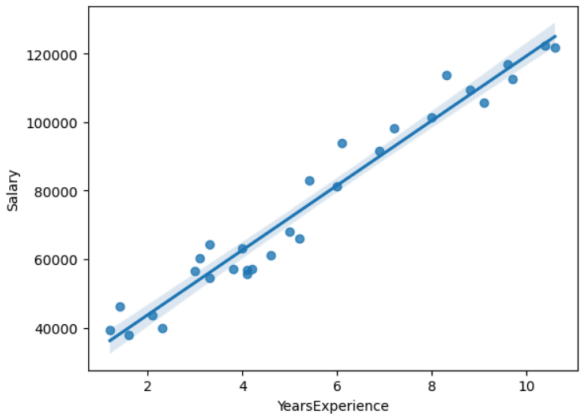
```
y_preddd = model_reg.predict(X)
```

```
print(model_reg.predict([[1.6]]))
```

```
[39968.14368085]
C:\Users\kalat\anaconda3\Lib\site-packages\sklearn\utils\v
itted with feature names
  warnings.warn(
```

```
plt.scatter(X, y, color='blue', label='Data points')
plt.plot(X, y_pred, color='red', label='Regression line')
plt.xlabel('Independent Variable (X)')
plt.ylabel('Dependent Variable (y)')
plt.title('Simple Linear Regression')
plt.legend()
plt.show()
```

```
sns.regplot(x = X, y = y)
plt.show()
```



Logistic Regression

Overview :

Logistic Regression is a statistical model that is used for classification problems, particularly for binary classification. Despite the name, it's a classification algorithm rather than a regression algorithm. It models the relationship between a dependent binary variable (target variable) and one or more independent variables (predictors/features). The output of logistic regression is a probability that a given input point belongs to a particular class.

Key Points:

- Binary Classification: Logistic Regression is typically used for problems where the output variable has two classes (i.e., binary classification).
- Sigmoid Function: Logistic regression uses the sigmoid function to map predicted values to probabilities in the range of 0 and 1.
- Linear Combination: The model assumes that a linear combination of the input features (weighted by coefficients) will predict the output. This combination is passed through a sigmoid function to get the probability.

Use Cases of Logistic Regression:

- Email Spam Detection: Predicting whether an email is spam or not (binary outcome: spam or not spam).
- Credit Card Fraud Detection: Identifying whether a transaction is fraudulent or not (fraud or non-fraud).
- Medical Diagnosis: Predicting whether a patient has a certain disease or not (positive or negative diagnosis).
- Customer Churn Prediction: Predicting whether a customer will stay or leave a service (churn or not churn).

Sigmoid Function (Why it is Used):

- Range [0, 1]: The sigmoid function maps any real-valued number into the range between 0 and 1. This is useful because the output of the model represents the probability of belonging to the positive class.
- Smooth Output: Unlike linear regression, which can output any value (positive or negative), the sigmoid function ensures the output is always between 0 and 1, making it perfect for classification.

Binary Classification

Binary Classification is a type of classification task where the goal is to predict one of two possible outcomes or classes. The output variable (target) in binary classification has only two possible values, often represented as 0 and 1, True and False, or Positive and Negative.

Sigmoid Function is used in logistic regression for binary classification.

```
X_diabetes = data_diabetes[['Age', 'Gender','BMI', 'Family_History', 'Physical_Activity','Diet_Type', 'Smoking_Status',
                            'Alcohol_Intake', 'Stress_Level', 'Hypertension', 'Glucose_Tolerance_Test_Result', 'Vitamin_D_Level', 'C_Protein_Level']]
y_diabetes = data_diabetes['Diabetes_Status']
```

```
X_train, X_test, y_train, y_test = train_test_split(X_diabetes, y_diabetes, test_size = 0.3)
```

```
model_logistic = LogisticRegression()
model_logistic.fit(X_train, y_train)
```

```
C:\Users\kalat\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression

```
LogisticRegression()
```

```
y_pred_logistic = model_logistic.predict(X_test)
```

```
print('Accuracy              :', accuracy_score(y_test, y_pred_logistic))
print('Confusion Matrix      :\n', confusion_matrix(y_test, y_pred_logistic))
print('Classification Report :\n', classification_report(y_test, y_pred_logistic))
```

```
Accuracy              : 0.49748110831234255
Confusion Matrix      :
 [[272 512]
 [286 518]]
Classification Report :
              precision    recall  f1-score   support

          No       0.49      0.35      0.41       784
         Yes       0.50      0.64      0.56       804

    accuracy                           0.50      1588
   macro avg       0.50      0.50      0.49      1588
weighted avg       0.50      0.50      0.49      1588
```

Decision Trees

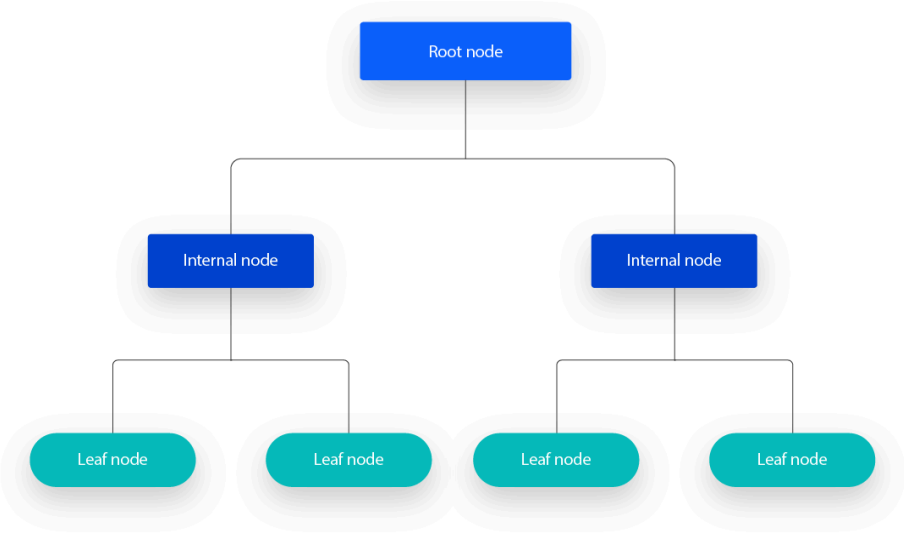
Decision Trees are a popular and powerful tool in the realm of machine learning, widely used for both classification and regression tasks. Their intuitive structure, mirroring human decision-making processes, makes them highly interpretable and easy to understand.

What is a Decision Tree?

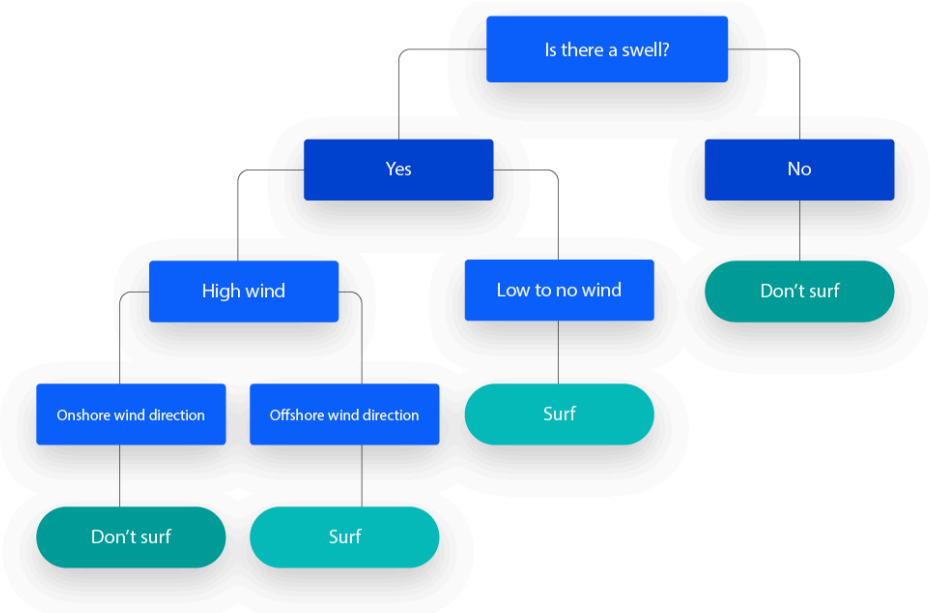
A Decision Tree is a flowchart-like structure where each internal node represents a decision based on the value of an attribute, each branch represents the outcome of a decision, and each leaf node represents a class label (in classification) or a continuous value (in regression). The paths from the root to the leaf represent classification rules or regression predictions.

Components of a Decision Tree

- Root Node: The topmost node, representing the entire dataset, which is split into two or more homogeneous sets.
- Decision Nodes: Nodes where the data is split based on certain attributes.
- Leaf/Terminal Nodes: Nodes that represent the final output or prediction; no further splits occur beyond these points.
- Branches: Paths connecting nodes, representing outcomes of a decision rule applied to an attribute.



As an example, let's imagine that you were trying to assess whether or not you should go surf, you may use the following decision rules to make a choice :



Splitting Criteria

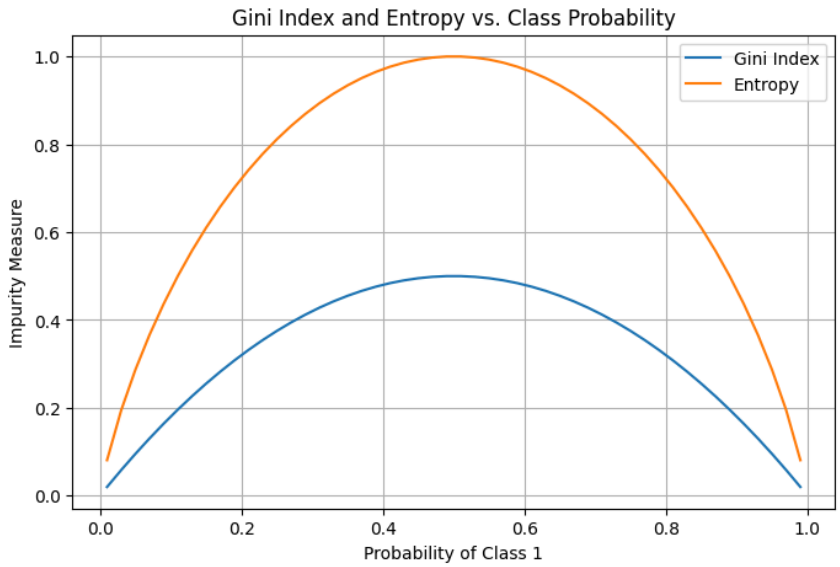
When constructing a Decision Tree, especially when there are more than two options at each split, it is crucial to determine which attribute to use at each internal node. This decision is guided by criteria that measure how well an attribute splits the data into homogeneous subsets. The two most common criteria are:

1. Gini Impurity: Measures the frequency at which any element of the dataset would be mislabeled if it was randomly labeled according to the distribution of labels in the subset. Lower Gini impurity indicates a better split.

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

2. Information Gain: Based on the concept of entropy from information theory, it measures the reduction in entropy after a dataset is split on an attribute. Higher Information Gain indicates a better split.

$$\text{Entropy}(S) = -\sum_{c \in C} p(c) \log_2 p(c)$$



Use Cases of Decision Trees:

1. <u>Classification:</u>
    - Medical Diagnosis: Classifying whether a patient has a disease based on symptoms.
    - Credit Scoring: Predicting whether an individual will default on a loan (e.g., yes/no).
    - Email Spam Detection: Classifying emails as spam or not spam.
    - Customer Segmentation: Classifying customers based on purchasing behavior (high-value vs low-value).
2. <u>Regression:</u>
    - House Price Prediction: Predicting the price of a house based on features like size, location, etc.
    - Stock Price Forecasting: Predicting future stock prices based on historical data.
    - Weather Forecasting: Predicting temperature or rainfall based on weather patterns.

```
columns_to_label = ['Gender', 'Family_History', 'Physical_Activity', 'Diet_Type', 'Smoking_Status', 'Alcohol_Intake',
                    'Stress_Level', 'Hypertension', 'Urban_Rural', 'Health_Insurance', 'Regular_Checkups', 'Medication_For_Chronic_Conditions',
                    'Polycystic_Ovary_Syndrome', 'Thyroid_Condition']

for column in columns_to_label :
    data_diabetes_dt[column] = label_encoder.fit_transform(data_diabetes_dt[column])
```

```
X_dt = data_diabetes_dt.drop("Diabetes_Status", axis=1)
y_dt = data_diabetes_dt["Diabetes_Status"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X_dt, y_dt, test_size = 0.3)
```

```
model_dt = DecisionTreeClassifier(criterion = 'entropy', random_state = 27, max_depth = 5)
model_dt.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=27)
```

```
y_pred_dt = model_dt.predict(X_test)
```

```
print('Accuracy              :', accuracy_score(y_test, y_pred_dt))
print('Confusion Matrix      :\n', confusion_matrix(y_test, y_pred_dt))
print('Classification Report :\n', classification_report(y_test, y_pred_dt))
```

```
Accuracy              : 0.48614609571788414
Confusion Matrix      :
 [[167 607]
 [209 605]]
Classification Report :
              precision    recall  f1-score   support

          No       0.44      0.22      0.29       774
         Yes       0.50      0.74      0.60       814

    accuracy                           0.49      1588
   macro avg       0.47      0.48      0.44      1588
weighted avg       0.47      0.49      0.45      1588
```

```
plt.figure(figsize=(20, 10))
plot_tree(model_dt, feature_names=X_dt.columns, class_names=["No Diabetes", "Diabetes"], filled=True)
plt.show()
```