10/03/25 Iterative Dichotomiser 3 (ID3)

* The ID3 algorithm is specifically designed for building decision trees from a given dataset. Its primary objective is to construct a tree that best explains the relationship between attributes in the data and their corresponding class labels.

ID3 Algorithm

1. Input
- A set of training examples, each with a target class.
- A set of features with possible values (discrete attributes)

2. Initialization
- Start with the entire dataset as the root node

3. Recursion: For each node:

(i) Base Cases:
- If all instances in the dataset belong to the same class, create a leaf node with that class
- If there are no features left to split on, create a leaf node with the majority class.

PAGE NO :
DATE :

(ii) Calculate Entropy : For the dataset at the current node, calculate the entropy

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i is the probability of class i in the dataset S.

(iii) Calculate Information gain. For each feature, calculate the information gain

$$IG(S,A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

where S_v is the subset of the data where the feature A takes the value v

(iv) Select the Feature with the Highest Information gain

4. Split the Dataset: Partition the dataset into subsets based on the selected feature that For each subset:

- Recursively apply the ID3 algorithm to the subset, treating it as

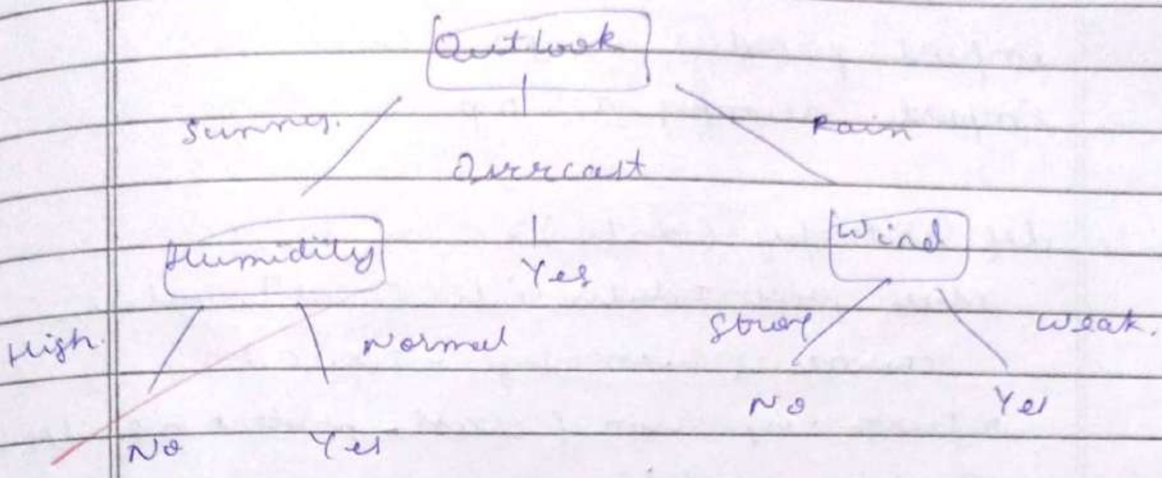
5. Construct the Tree: Repeat 3 and 4 recursively for each node until the stopping condition is met.

6. Output:

- A decision tree representing the learned classification model.

PAGE NO :
DATE :

Decision tree is:



10/3/25

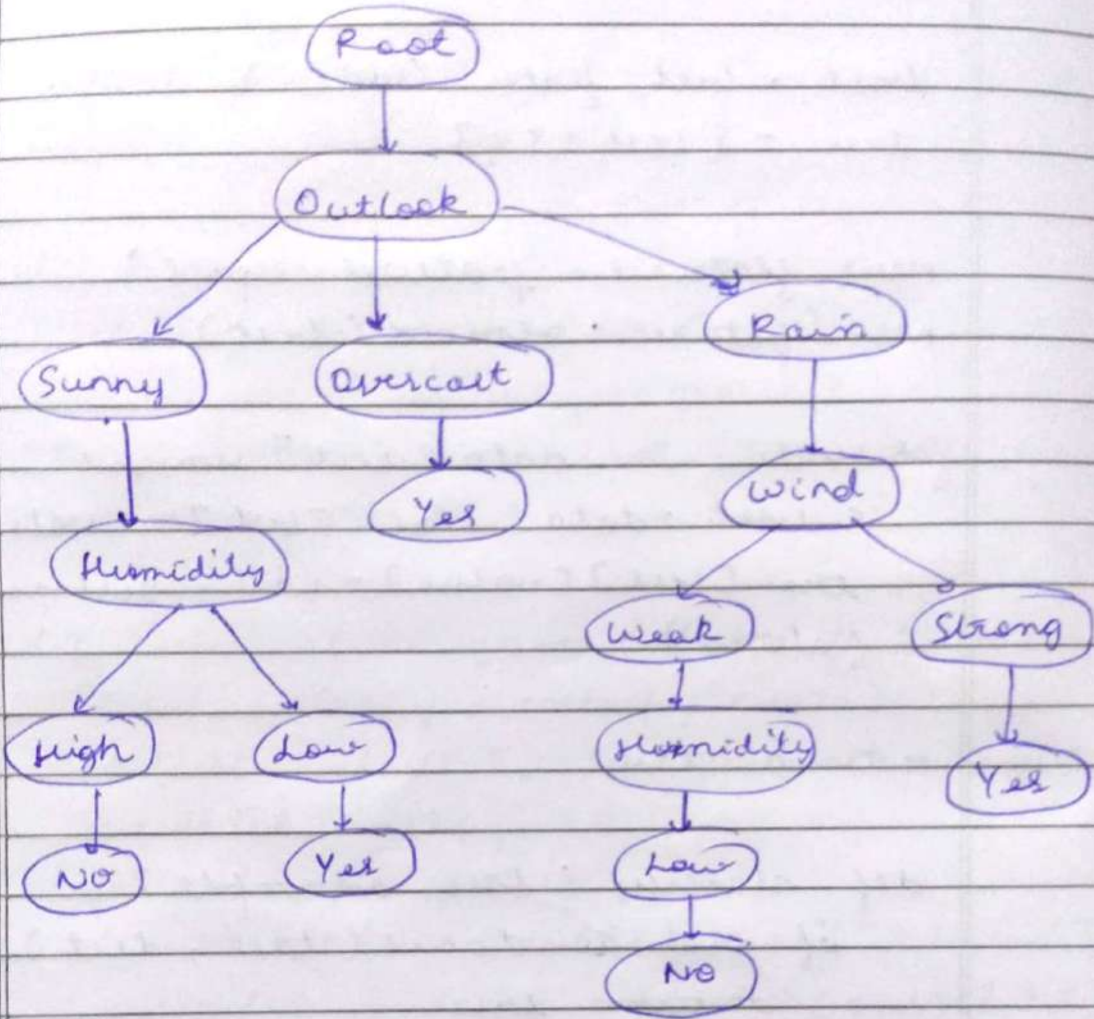
17/03/24 ID3 code :

```
import pandas as pd
import numpy as np

def entropy (data):
    class_prob = data.iloc[:, -1].value_
      counts (normalize =True)
    return -np.sum (class_prob * np.log2
     (class_prob))

def information_gain (data, feature) :
    Total_entropy = entropy (data)
    feature_values = data[feature].unique()
    weighted_entropy =0
    for value in feature_values:
      subset = data[data[feature] == value]
      weighted_entropy += (len (subset) /
       len (data)) * entropy (subset)
    return total_entropy - weighted_entropy

def best_feature (data):
    features = data.columns [:-1]
    gains = {feature: information_gain(
     data, feature) for feature in features }
    return max(gains, key = gains.get)

def id3 ( data, features =None ):
    if len (data.iloc [:,-1].unique ())==1:
      return data.iloc [:,-1].iloc[0]

    if len ( features)==0 :
```

PAGE NO :
DATE :

```
return data.iloc[:,-1].mode()[0]

best = best_feature(data)
tree = {best: {}}

new_features = features.copy()
new_features.remove(best)

for value in data[best].unique:
    subset = data[data[best] == value]
    tree[best][value] = id3(subset, new_
    features)

return tree

def classify(tree, example):
    if not isinstance(tree, dict):
        return tree
    feature = list(tree.keys())[0]
    value = example[feature]
    return classify(tree[feature][value], exampl

data = pd.read_csv("/3-dataset.csv")

tree = id3(data, features = list(data.columns
                                        [:-1]))
print("Decision tree:", tree)

example = {'Outlook': 'Sunny', 'Temperature':
'Cool', 'Humidity': 'Low', 'Wind': Strong}
prediction = classify(tree, example)
print("Prediction for the example:",
      prediction)
```

Decision tree :



Prediction for the example : no

PAGE NO :
DATE :

* End to end machine learning project working with real data. Look at the big picture, visualize the data. Prepare the data, select and train the model and fine tune it.

1. Get the data

```
import pandas as pd
housing = pd.read_csv("sample_data/
california_housing_train.csv")
```

2. Discover the data

```
housing.head()
housing.info()
housing.describe()
```

3. Visualize the data

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.hist(housing['median_income'])
plt.show()
plt.scatter(housing['median_income'],
housing['median_house_value'])
plt.show()

sns.heatmap(housing.corr(), annot=True)
plt.show()
```

PAGE NO :
DATE :

4. Prepare the data

```
housing.isnull().sum()
```

5. Select and train the model

```
from sklearn.model_selection
import train_test_split
from sklearn.preprocessing import
OneHotEncoder

X = housing.drop('median_house_value',
axis = 1)
y = housing['median_house_value']
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size = 0.2,
random_state = 42)

from sklearn.linear_model import
LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

6. Fine tune your model

```
from sklearn.metrics import root_mean_squared_error
import numpy as np

y_pred = model.predict(X_test)
rmse = root_mean_squared_error(y_test, y_pred)
print(f'RMSE: {rmse}')
```