17/3/2025 Lab 3 Implementation of ID3 Algorithm

```
import pandas as pd
import numpy as np

def entropy(data):
    class_probabilities = data.iloc[:,-1].
                          value_counts(normalize=
                          True)
    return -np.sum( class_probabilities *
                    np.log2(class_probabilies)

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 t= (len(subset) /
                             len(data)) *
                            entropy(subset)
    return total_entropy - weight_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].mode()[0]

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

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

    for value in dat[best].unique():
        subset = data[data[best] == value]
        tree[best][value] = id3(subset, new_features)
    return tree.
```

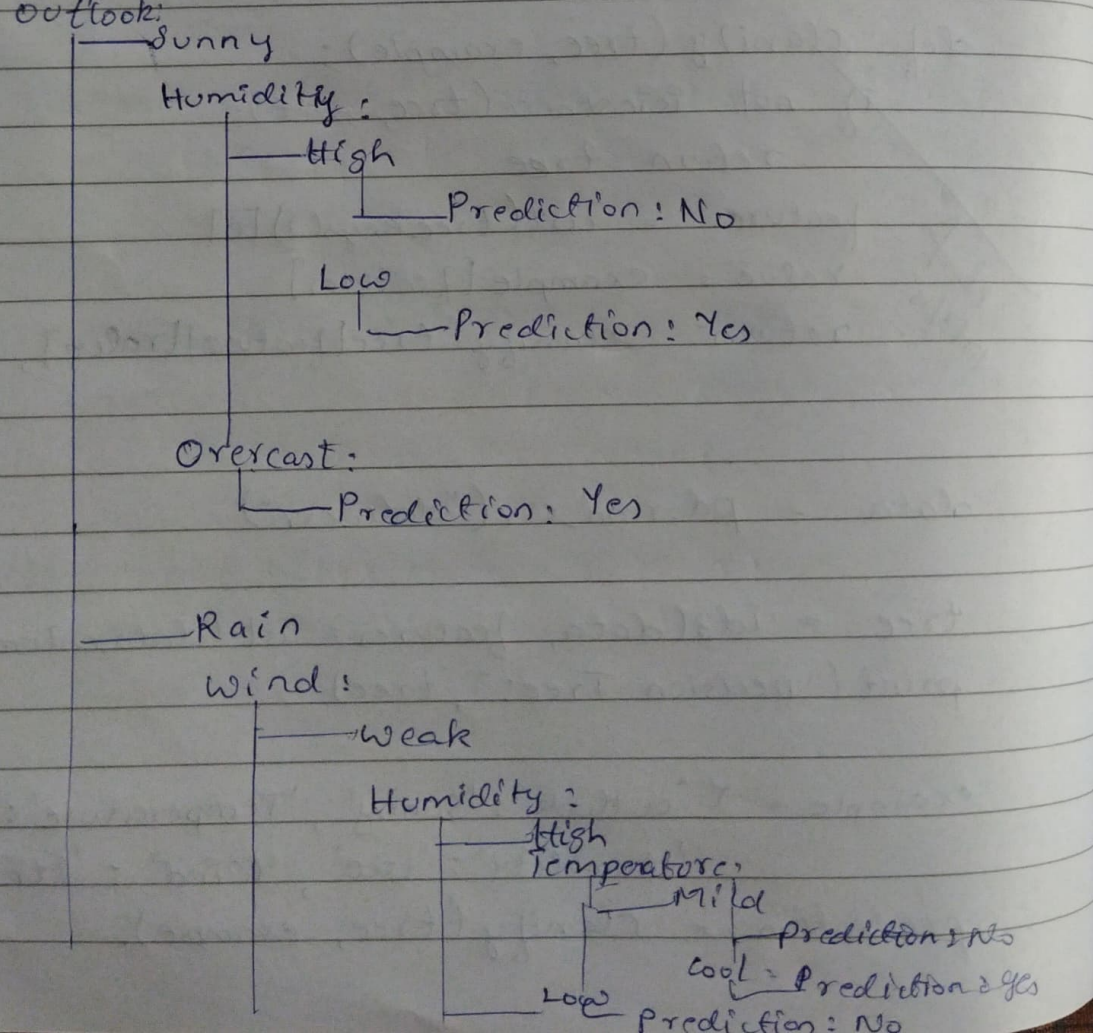
```
def classify(tree, example):
    if not isinstance(tree, dict):
        return tree
    features = list(tree.keys())[0]
    value = example[features]
    return classify(tree[feature][value], examply)
```

17/3/25

```
data = pd.read_csv("data.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)

def print_tree(tree, indent=" "):
    if isinstance(tree, dict):
        for key, value in tree.items():
            print(f"{indent}{key} :")
            for sub_key, sub_tree in value.items
                print(f"{indent}. {sub_key}")
                print_tree(sub_tree, indent + "|
    else:
        print(f"{indent}  Prediction: {tree}")
```

output:



Strong

Prediction: Yes

Prediction for example: Yes

* End to end machine learning project working with null data. Look at the big picture get the data discover. Visualize the data Program the data, select and train the model ~~for~~ fine tune your model.

1) Get the data

```
Import pandas as pd
housing = pd.read_csr("/sample_data/california.csr")
```

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'])
plt.show()

plt.scatter(house['median_income'], housing['median_
                                    house_value'])
```

121

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

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 OneHotEnco

x = housing.drop('median_house_value', axis=1)
y = housing['median_house_value']

x_train, x_test, y_train, y_test = train_test_spli
        (x, y, test size = 0.2, random_stat
                                    =42)

from sklearn linear_model import
                        Linear Regression
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}')
```

12/3/25