reat on the selected feature
or each unique value of
and partition the
- for each branch.
ID3 to each subset
ID3 algorithm to
dataset until some
n is met.

tree is built, it can
g branches that
ically using data.

M T W T F S S
Page No.:
Date:
YOUVA

LAB-3

ID3 code.

code:-

```
import numpy as np
import pandas as pd.
from graphviz import Digraph.
def entropy(dataset):
    class_counts = dataset.iloc[:,-1].value_counts()
    prob = class_counts / len(dataset)
    return -np.sum(prob * np.log2(prob))
def info_gain(dataset, feature):
    tot_entropy = entropy(dataset)
    feature_value = dataset[feature].value_counts()
    weighted_entropy = 0
    for value, count in feature_value.items():
        subset = dataset[dataset[feature] == value]
        weighted_entropy = (count / len(dataset)) *
                           entropy(subset)
    return total_entropy - weighted_entropy
def best_feature(dataset):
    feature = dataset.columns[:-1]
    best_info_gain = -1
    best_feature = None
    for feature in feature:
        info_gain = information_gain(dataset, feature)
        if info_gain > best_info_gain:
            best_info_gain = info_gain
```

```
best_feature = feature
return best_feature
def id3(dataset, max_depth=None, depth=0):
    if len(dataset.iloc[:, -1].unique()) == 1:
        return dataset.iloc[0, -1]
    if len(dataset.columns) == 1:
        return dataset.iloc[:, -1].mode()[0]
    best = best_feature(dataset)
    tree = {best: {}}
    for value in dataset[best].unique():
        subset = dataset[dataset[best] == value]
def create_tree_diagram(tree, dot=None,
parent="Root", parent_value=""):
    if dot is None:
        dot = Digraph(format="png", engine="dot")
    if isinstance(tree, dict):
        for feature, branches in tree.items():
            feature_name = f"{parent_name}_{feature}"
            dot.node(feature_name, feature)
            dot.edge(parent_name, feature_name,
            label=parent_value)
else:
    dot.node(parent_name + "_leaf", f"Class {tree}")
    dot.edge(parent_name, parent_name + "_leaf",
             label="Leaf")
return dot
```

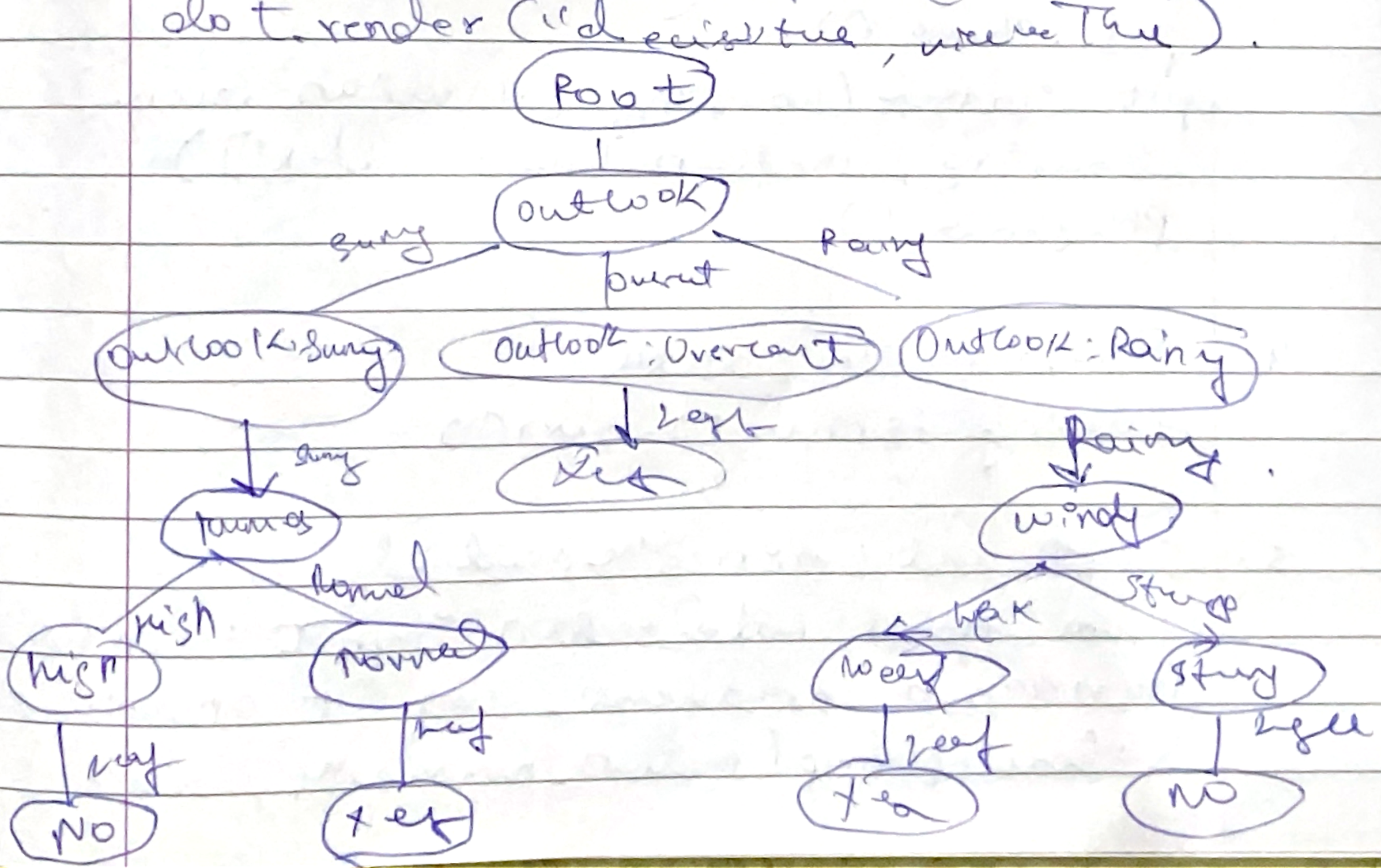
data = {'outlook': [Sunny, Sunny, Overcast, Rainy, Rainy, Rainy, Overcast, Sunny, Sunny, Rainy, Sunny, Overcast, Overcast, Rainy],
'Temp': [Hot, Hot, Hot, Mild, Cool, Cool, Mild, Cool, Mild, Mild, Mild, Hot, Mild],
'Humidity': [High, High, High, High, Normal, Normal, Normal, High, Normal, Normal, Normal, High, Normal, High],
'Windy': [Weak, Strong, Weak, Weak, Weak, Strong, Strong, Weak, Weak, Weak, Strong, Strong, Weak, Strong],
'Playtennis': [No, No, Yes, Yes, Yes, No, Yes, No, Yes, Yes, Yes, Yes, Yes, No]}

df = pd.DataFrame(data)

tree = id3(df, max_depth=3)

dot = create_tree_diagram(tree)

dot.render("decision_tree", view=True).



To DO:- END to END Machine learning project

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()
```

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 ('media_house_value')
x_train, x_test, y_train, y_test : train_tst,
split (x, y, testsize = 0.2, random_state = 49)
from sklearn.linear_model import Linear Regression
model = Linear Regression()
model.fit(x_train, y_train)

b. 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}")

17/3/25