jupyter r17cs159-mldal-Prog3 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```
        elif v[0] != 0 or v[1] != 0:
            impure_childs.append(k)
    for i in impure_childs:
        sub = modify_data_set(dataset,attribute_index, dataset.columns[attribute_index], i)
        tree = construct_tree(sub, tree)
    return tree


# ---------------------------------------------------------------------


# main function
# ---------------------------------------------------------------------
def main():
    df = pd.read_csv("data1.csv")
    tree = dict()
    result = construct_tree(df, tree)
    for key, value in result.items():
        print(key, " => ", value)


# ---------------------------------------------------------------------

if __name__ == "__main__":
    main()
```

```
Outlook  =>  {'Sunny': (5, 0), 'Rainy': (2, 3)}
Sunny  =>  Yes
Company  =>  {'No': (0, 2), 'Medium': (1, 1), 'Big': (1, 0)}
No  =>  No
Big  =>  Yes
Sailboat  =>  {'Small': (1, 0), 'Big_S': (0, 1)}
Small  =>  Yes
Big_S  =>  No
```

jupyter r17cs159-mldal-Prog3 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```
    attribute_index = greatest_information_gain(dataset)
    childs = generate_childs(dataset, attribute_index)
    tree[dataset.columns[attribute_index]] = childs
    targets = list(set(dataset.iloc[:, -1]))
    for k, v in childs.items():
        if v[0] == 0:
            tree[k] = targets[1]
        elif v[1] == 0:
            tree[k] = targets[0]
        elif v[0] != 0 or v[1] != 0:
            impure_childs.append(k)
    for i in impure_childs:
        sub = modify_data_set(dataset,attribute_index, dataset.columns[attribute_index], i)
        tree = construct_tree(sub, tree)
    return tree


# ---------------------------------------------------------------------


# main function
# ---------------------------------------------------------------------
def main():
    df = pd.read_csv("data2.csv")
    tree = dict()
    result = construct_tree(df, tree)
    for key, value in result.items():
        print(key, " => ", value)


# ---------------------------------------------------------------------

if __name__ == "__main__":
    main()
```

```
Outlook  =>  {'Sunny': (3, 2), 'Overcast': (0, 4), 'Rain': (2, 3)}
Overcast  =>  Yes
Humdity  =>  {'High': (3, 0), 'Normal': (0, 2)}
High  =>  No
Normal  =>  Yes
Wind  =>  {'Weak': (0, 3), 'Strong': (2, 0)}
Weak  =>  Yes
Strong  =>  No
```