```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor

df=pd.read_csv("/content/drive/MyDrive/food/Dataset.csv")
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 45584,\n  \"fields\":
[\n    {\n      \"column\": \"ID\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 45584,\n
\"samples\": [\n          \"0xcfcd\",\n          \"0x4f03\",\n
\"0xc48b\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Delivery_person_ID\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1320,\n
\"samples\": [\n          \"LUDHRES02DEL03\",\n
\"VADRES02DEL01\",\n          \"INDORES13DEL01\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Delivery_person_Age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
5.815063854587488,\n        \"min\": 15.0,\n        \"max\": 50.0,\n
\"num_unique_values\": 22,\n        \"samples\": [\n          36.0,\n
26.0,\n          25.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Delivery_person_Ratings\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.3347437201745157,\n
\"min\": 1.0,\n        \"max\": 6.0,\n        \"num_unique_values\":
28,\n        \"samples\": [\n          3.5,\n          2.8,\n
4.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Restaurant_latitude\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 8.185673779879592,\n        \"min\": -
30.905562,\n        \"max\": 30.914057,\n
\"num_unique_values\": 657,\n        \"samples\": [\n          -
17.426228,\n          12.981615,\n          17.431668\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Restaurant_longitude\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
22.8855748639992,\n        \"min\": -88.366217,\n        \"max\":
88.433452,\n        \"num_unique_values\": 518,\n        \"samples\":
[\n          -73.804855,\n          88.433187,\n          76.654878\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Delivery_location_latitude\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7.335561662334285,\n        \"min\": 0.01,\n        \"max\":
```

```
31.054057,\n        \"num_unique_values\": 4373,\n        \"samples\":
[\n          27.29185,\n          18.583811,\n          11.076686\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Delivery_location_longitude\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
21.12057827510338,\n        \"min\": 0.01,\n        \"max\":
88.563452,\n        \"num_unique_values\": 4373,\n        \"samples\":
[\n          78.170165,\n          73.949315,\n          77.021736\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Order_Date\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 44,\n        \"samples\": [\n          \"16-02-
2022\",\n          \"06-04-2022\",\n          \"04-04-2022\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Time_Orderd\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 176,\n
\"samples\": [\n          \"22:35\",\n          \"22:25\",\n
\"17:40\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Time_Order_picked\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 193,\n
\"samples\": [\n          \"09:40\",\n          \"11:05\",\n
\"10:40\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Weather_conditions\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          \"Fog\",\n          \"Stormy\",\n          \"Sunny\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Road_traffic_density\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"High\",\n          \"Low\",\n          \"Jam\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Vehicle_condition\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          1,\n
3,\n          2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Type_of_order\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Meal\",\n          \"Buffet\",\n          \"Snack\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Type_of_vehicle\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"scooter\",\n          \"bicycle\",\n          \"motorcycle\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
```

```
}\n    },\n    {\n      \"column\": \"multiple_deliveries\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.5725098447367195,\n        \"min\": 0.0,\n        \"max\": 3.0,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          1.0,\n
2.0,\n          3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Festival\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Yes\",\n          \"No\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"City\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Metropolitian\",\n          \"Urban\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Time_taken (min)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
9,\n        \"min\": 10,\n        \"max\": 54,\n
\"num_unique_values\": 45,\n        \"samples\": [\n          48,\n
16\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df = pd.DataFrame(df)

le = LabelEncoder()
df['Road_traffic_density'] =
le.fit_transform(df['Road_traffic_density'])
df['Weather_conditions'] = le.fit_transform(df['Weather_conditions'])

le = LabelEncoder()
df['Type_of_order'] = le.fit_transform(df['Type_of_order'])
X = df[['Weather_conditions', 'Road_traffic_density',
'Vehicle_condition', 'Type_of_order']]
y = df['Time_taken (min)']

model = RandomForestRegressor(n_estimators=5, random_state=42)
model.fit(X, y)
```

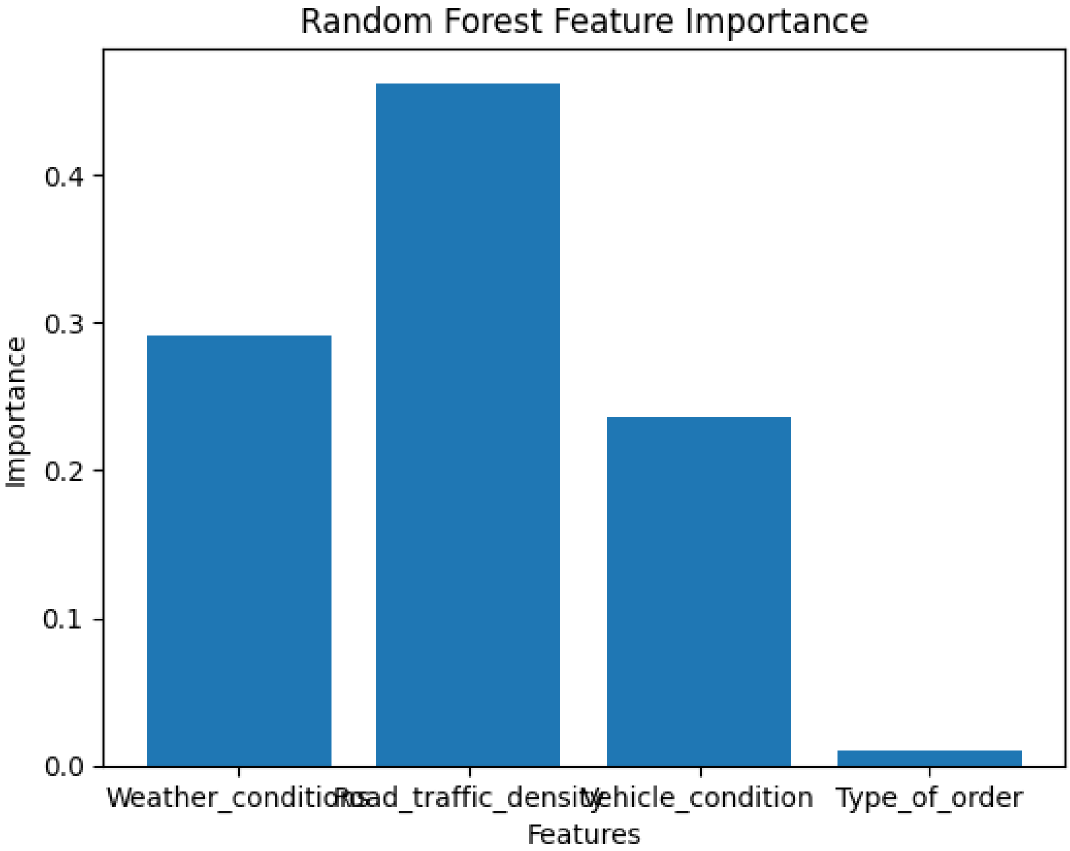
```
RandomForestRegressor(n_estimators=5, random_state=42)
```

```
pred = model.predict([[4.5, 5, 1, 2]])
print("Predicted Delivery Time:", pred)
```

```
Predicted Delivery Time: [17.72910507]
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but RandomForestRegressor was fitted with feature names
  warnings.warn(
importance = model.feature_importances_

plt.figure()
plt.bar(X.columns, importance)
plt.xlabel("Features")
plt.ylabel("Importance")
plt.title("Random Forest Feature Importance")
plt.show()
```



```
tree = model.estimators_[0]

plt.figure(figsize=(20,10))
plot_tree(
    tree,
    feature_names=X.columns,
    filled=True,
    rounded=True
)
plt.title("Random Forest – One Decision Tree Visualization")
plt.show()
```

Random Forest – One Decision Tree Visualization