COGNIFYZ MACHINE LEARNING INTERNSHIP-RESTAURANT ANALYSIS

DHULIPUDI VEERA VENKATA SWAMINADH

RESTAURANT PREDICTION

ABOUT DATA

- The data contains restaurant id, restaurant name, country code, location, aggregate rating, rating, locality, city, address, longitude, latitude, phone number, currency, rating text, voting etc…

TASKS

- Predict Restaurant Ratings
- Restaurant Recommendation
- Cuisine Classification
- Location Based Analysis

TASK1 – PREDICT RESTAURANT RATINGS

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
df = pd.read_csv("restaurant_data.csv")
columns_list = df.columns.tolist()
print(columns_list)
print(df.dtypes)
X = df.drop("Aggregate rating", axis=1)
y = df["Aggregate rating"]
non_numeric_columns = df.select_dtypes(exclude=['float64', 'int64']).columns
print("Non-numeric columns:", non_numeric_columns)
X_encoded = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2,
random_state=42)
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
```


Inadequate
Requires improvement
Good
Outstanding
Percentage
0
20
40
60
80
100
Overall
Caring
Effective
Responsive
Safe
Well led
Key domain

PROCEDURE

Embark on a journey into the realm of machine learning optimization by delving into the art of fine-tuning a Random Forest Classifier using scikit-learn. Discover the art of preprocessing data, strategically dividing it into training and testing sets, and unearthing the elusive optimal hyperparameters to elevate your model's performance. Gain profound insights into how distinct parameter configurations impact accuracy, precision, recall, and the F1-score. This voyage offers a hands-on encounter with real-world data analysis, equipping you with the skills to master the art of model refinement

[4]
```
#print the columns names
print("Columns in the dataset:")
print(columns_list)
```

```
Columns in the dataset:
['Restaurant ID', 'Restaurant Name', 'Country Code', 'City', 'Address', 'Locality', 'Locality Verbose', 'Longitude', 'Latitude', 'Cuisines',
```

[5]
```
#print datatypes
print(df.dtypes)
```

```
Restaurant ID              int64
Restaurant Name           object
Country Code               int64
City                      object
Address                   object
Locality                  object
Locality Verbose          object
Longitude                float64
Latitude                 float64
Cuisines                  object
Average Cost for two       int64
Currency                  object
Has Table booking         object
Has Online delivery       object
Is delivering now         object
Switch to order menu      object
Price range                int64
Aggregate rating         float64
Rating color              object
Rating text               object
Votes                      int64
dtype: object
```

```
#Decision trees allow you to interpret feature importance easily. You can analyze the feature_importances_ attribute of the model.
importance = model.feature_importances_
feature_names = X_encoded.columns
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importance})
sorted_features = feature_importance_df.sort_values(by='Importance', ascending=False)

print("\nMost Influential Features:")
print(sorted_features)
```

```
Most Influential Features:
                              Feature    Importance
20825           Rating text_Not rated  8.966539e-01
20819          Rating color_Orange  5.152611e-02
20826              Rating text_Poor  2.219755e-02
20824              Rating text_Good  1.308405e-02
20827         Rating text_Very Good  2.579984e-03
...                               ...           ...
17386                Locality_Saket -1.008831e-19
16589              Locality_Augusta -2.017661e-19
2871   Restaurant Name_Harichatni.com -2.017661e-19
5379         Restaurant Name_Rollmaal -3.026492e-19
260          Restaurant Name_Alaturka -3.026492e-19

[20828 rows x 2 columns]
```

TASK2 – RESTAURANT RECOMMENDATION

SOURCE CODE

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
df = pd.read_csv("restaurant_data.csv")
predicted_rating =
model.predict([[sample_user_pref_encoded]])[0]
print(f"Predicted Rating for User Preferences:
{sample_user_preferences}: {predicted_rating:.2f}")
```

PROCEDURE

The code processes cuisine data, constructs a Decision Tree Regressor for rating predictions, assesses performance, and illustrates predicting user preferences. This highlights core regression steps using scikit-learns capabilities

```
print("Mean Squared Error (MSE):", mse)
print("R-squared (R2):", r2)
```

```
Mean Squared Error (MSE): 1.941493078539205
R-squared (R2): 0.1470122534531093
```

```
# Make a prediction for sample user preferences
sample_user_preferences = 'Italian, Chinese'  # Sample user preferences for cuisine
sample_user_preferences = sample_user_preferences.lower()
sample_user_pref_encoded = le.transform([sample_user_preferences])[0]
```

```
predicted_rating = model.predict([[sample_user_pref_encoded]])[0]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but D
  warnings.warn(
```

```
print(f"Predicted Rating for User Preferences: {sample_user_preferences}: {predicted_rating:.2f}")
```

```
Predicted Rating for User Preferences: italian, chinese: 0.00
```

TASK3 – CUISINE CLASSIFICATION

SOURCE CODE

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
np.random.seed(42)
num_samples = 1000
num_features = 5
X = np.random.randn(num_samples, num_features)
y = np.random.randint(2, size=num_samples)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
y_pred = logistic_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
print("Model Performance:")
print("Accuracy:", accuracy)
print("Precision:", precision)
```

PROCEDURE

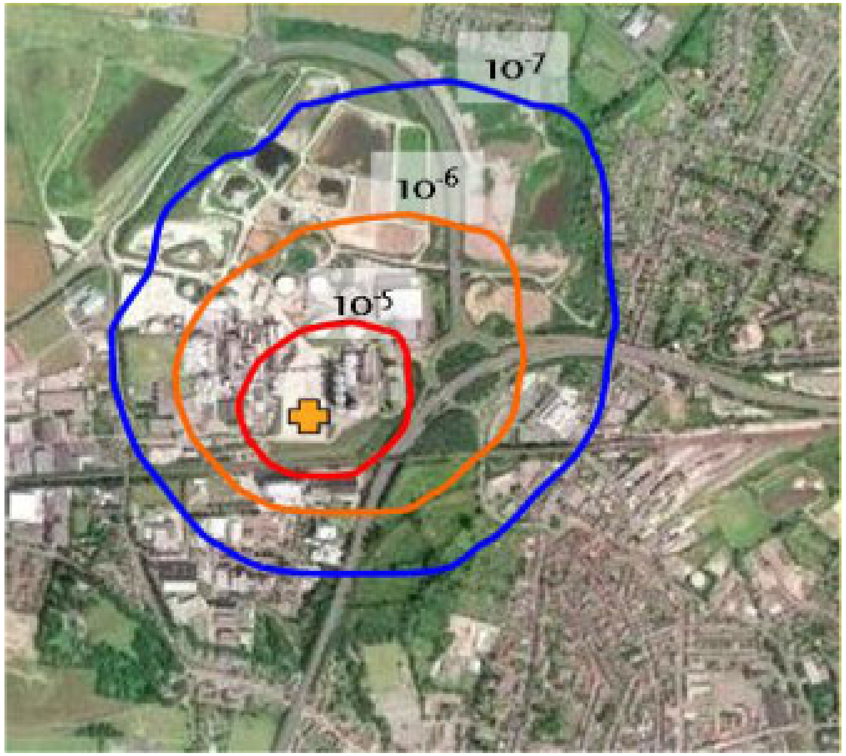
Code demonstrates Logistic Regression for binary classification. Synthetic data is created, split into train/test sets, model is trained and assessed using accuracy, precision, recall, F1 score, and confusion matrix. Illustrates model training and evaluation in classification

```python
#Evaluate the model's performance using confusion metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
confusion = confusion_matrix(y_test, y_pred)

print("Model Performance:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confuion Matrix:\n", confusion)
```

```
Model Performance:
Accuracy: 0.48
Precision: 0.47874188311688315
Recall: 0.48
F1 Score: 0.47817103808662437
Confuion Matrix:
 [[41 57]
 [47 55]]
```

TASK4 – LOCATIO N BASED ANALYSIS



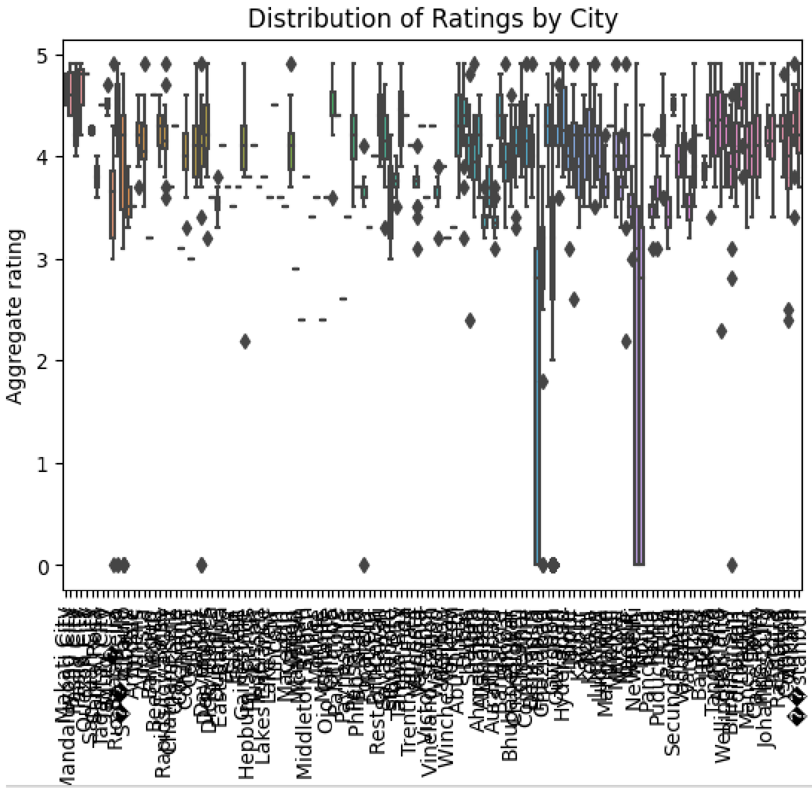
SOURCE CODE

```
import pandas as pd
import folium
from folium import plugins
df = pd.read_csv("restaurant_data.csv")
map_restaurants.save("restaurant_map.html")
restaurant_count_by_city =
df.groupby('City').size().reset_index(name='Restaurant Count')
sns.boxplot(x='City', y='Aggregate rating',data=df)
plt.xticks(rotation=90)
plt.title('Distribution of Ratings by City')
plt.show()
```

Code utilizes Folium for interactive maps, Pandas for data analysis, Seaborn,Matplotlib for visualizations, creating a

PROCEDURE

It employs Folium for interactive map visualization, Pandas for data analysis, Seaborn and Matplotlib for data visualization, and combines all these components to create a comprehensive view of restaurant data. The code showcases both geographical and statistical insights into restaurant locations, counts, average ratings, and the distribution of ratings across different cities. This demonstrates a multi-faceted exploration and presentation of the restaurant dataset.



Distribution of Ratings by City
Aggregate rating
0
1
2
3
4
5

LINKS

- COLAB LINK – https://colab.research.google.com/drive/12ff3PamoGBY9IbjOxOOgWmNrC4JgXHD1?usp=sharing
- EMAIL –vvsnadh999@gmail.com
- LINKED IN – https://www.linkedin.com/posts/vvsnadh999_linkedin-kiet-machinelearning-activity-7115550120095436800-QGlp?utm_source=share&utm_medium=member_desktop


thank
you