exoplanet

October 4, 2024

[58]:

```
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier  # You can choose any␣
 ↪classifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

[11]:

```
df = pd.read_csv("PHL-EC.csv")
```

[12]:

```
df["P. Habitable Class"].value_counts()
```

[12]:

```
P. Habitable Class
non-habitable        3820
mesoplanet             31
psychroplanet          18
thermoplanet            3
hypopsychroplanet       3
Name: count, dtype: int64
```

[41]:

```
X = df.drop(columns=["P. Habitable Class","P. Max Mass (EU)"])
Y = df["P. Habitable Class"]
X = X.select_dtypes(include=['number'])
X.fillna(X.mean(), inplace=True)
```

[42]:

```
scaler  = StandardScaler()
X.isnull().sum()
x_scaled = scaler.fit_transform(X)
X.isnull().sum()
```

[42]:

```
P. Name KOI                0
P. Min Mass (EU)           0
P. Mass (EU)               0
```

```
P. Radius (EU)               0
P. Density (EU)              0
P. Gravity (EU)              0
P. Esc Vel (EU)              0
P. Teq Min (K)               0
P. Teq Mean (K)              0
P. Teq Max (K)               0
P. Ts Min (K)                0
P. Ts Mean (K)               0
P. Ts Max (K)                0
P. Surf Press (EU)           0
P. Mag                       0
P. Appar Size (deg)          0
P. Period (days)             0
P. Sem Major Axis (AU)       0
P. Eccentricity              0
P. Mean Distance (AU)        0
P. Inclination (deg)         0
P. Omega (deg)               0
S. Mass (SU)                 0
S. Radius (SU)               0
S. Teff (K)                  0
S. Luminosity (SU)           0
S. [Fe/H]                    0
S. Age (Gyrs)                0
S. Appar Mag                 0
S. Distance (pc)             0
S. RA (hrs)                  0
S. DEC (deg)                 0
S. Mag from Planet           0
S. Size from Planet (deg)    0
S. No. Planets               0
S. No. Planets HZ            0
S. Hab Zone Min (AU)         0
S. Hab Zone Max (AU)         0
P. HZD                       0
P. HZC                       0
P. HZA                       0
P. HZI                       0
P. SPH                       0
P. Int ESI                   0
P. Surf ESI                  0
P. ESI                       0
S. HabCat                    0
P. Habitable                 0
P. Hab Moon                  0
P. Confirmed                 0
```

```
Unnamed: 68                 0
dtype: int64
```

[43]:

```
pca = PCA(n_components=2)
x_pca = pca.fit_transform(x_scaled)
```

[94]:

```
pca_df  = pd.DataFrame(data = x_pca ,columns=["PC1","PC2"])
pca_df["P. Habitable Class"] = Y
pca_df.to_csv("PC_Exo.csv")
```

[47]:

```
df["P. Habitable Class"].value_counts()
```

[47]:

```
P. Habitable Class
non-habitable        3820
mesoplanet             31
psychroplanet          18
thermoplanet            3
hypopsychroplanet       3
Name: count, dtype: int64
```

[70]:

```
label_encoder = LabelEncoder()
Y = label_encoder.fit_transform(Y)
unique_classes = label_encoder.classes_
print("Unique classes in the target variable:", unique_classes)
```

```
Unique classes in the target variable: [0 1 2 3 4]
```

[50]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)
```

[83]:

```
# Initialize the model
model = RandomForestClassifier(n_estimators=100,␣
 ↪random_state=42,class_weight="balanced")

# Fit the model
model.fit(X_train, Y_train)
```

[83]:

```
RandomForestClassifier(class_weight='balanced', random_state=42)
```

[84]:

```
Y_pred = model.predict(X_test)
```

[85]:

```
confusion = confusion_matrix(Y_test, Y_pred)
print("Confusion Matrix:")
print(confusion)
```

```
Confusion Matrix:
[[  1   0   0   0]
 [  0   6   1   0]
 [  0   0 763   0]
 [  0   0   0   4]]
```

[90]:

```
report = classification_report(Y_test, Y_pred, output_dict=True)
```

[93]:

```
import pandas as pd
from pandas_profiling import ProfileReport
from pydantic_settings import BaseSettings
from sklearn.metrics import classification_report, accuracy_score

# Define your settings using Pydantic Settings
class Settings(BaseSettings):
    input_file: str = "report_exo.csv"  # Default value for the input file
    output_file: str = "Exoreport.html"  # Default value for the output file

    class Config:
        env_file = ".env"  # Optional: Load environment variables from a .env␣
 ↪file

# Initialize settings
settings = Settings()

# Calculate accuracy
accuracy = accuracy_score(Y_test, Y_pred)

# Generate classification report
report = classification_report(Y_test, Y_pred, output_dict=True,␣
 ↪zero_division=0)
report_df = pd.DataFrame(report).transpose()

# Add accuracy to the report
report_df['Metric'] = report_df.index
report_df['Accuracy'] = accuracy
report_df.to_csv("report_exo.csv")  # Save the classification report to the CSV␣
 ↪file

# Generate the profile report for the DataFrame
profile = ProfileReport(df, title='Data Profiling Report', explorative=True)

# Save the profiling report to HTML
profile.to_file(settings.output_file)

# Save the classification report to a separate HTML file
report_df.to_html("classification_report.html", index=False)

print(f"Profile report generated successfully: {settings.output_file}")
print("Classification report saved to classification_report.html.")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]

Profile report generated successfully: Exoreport.html
Classification report saved to classification_report.html.
```

[86]:

```python
# Calculate and print accuracy
accuracy = accuracy_score(Y_test, Y_pred)
accuracy_percentage = accuracy * 100  # Convert to percentage
print(f"Accuracy: {accuracy_percentage:.4f}%")
```

```
Accuracy: 99.8710%
```

[ ]: