26/11/2023

CSC461 – Assignment3 – Machine Learning

Numan Latif

FA21-BSE-039

Predict male and female using gender csv file and using machine learning

QUESTION 1

```
import pandas as pd

from google.colab import drive

drive.mount('/content/drive')

data = pd.read_csv('/content/drive/My Drive/IDS/Assignments/gender-prediction.csv')


num_instances = len(data)

print(f"1. Number of instances: {num_instances}")


num_input_attributes = len(data.columns)

print(f"2. Number of input attributes: {num_input_attributes}")


output_values = data['gender'].nunique()

print(f"3. Number of possible values for the output attribute: {output_values}")


categorical_attributes = data.select_dtypes(include=['object']).columns

num_categorical_attributes = len(categorical_attributes)

print(f"4. Number of categorical input attributes: {num_categorical_attributes}")


class_ratio = data['gender'].value_counts(normalize=True)

print(f"5. Class ratio (male and female):\n{class_ratio}")
```

```
print(f"5. Class ratio (male and female):\n{class_ratio}")

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
1. Number of instances: 110
2. Number of input attributes: 8
3. Number of possible values for the output attribute: 2
4. Number of categorical input attributes: 5
5. Class ratio (male and female):
male      0.563636
female    0.436364
Name: gender, dtype: float64
```

QUESTION 2

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X = data.drop("gender", axis=1)
y = data["gender"]

categorical_columns = X.select_dtypes(include=['object']).columns

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X.select_dtypes(include=['float64', 'int64']).columns),
        ('cat', OneHotEncoder(), categorical_columns)
    ]
)

lr_model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=42)

lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
```

```
incorrect_instances_lr = (y_test != y_pred_lr).sum()
print(f"1. Incorrect instances (Logistic Regression): {incorrect_instances_lr}")
X = data.drop("gender", axis=1)
y = data["gender"]
categorical_columns = X.select_dtypes(include=['object']).columns
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X.select_dtypes(include=['float64', 'int64']).columns),
        ('cat', OneHotEncoder(), categorical_columns)
    ]
)
lr_model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
incorrect_instances_lr = (y_test != y_pred_lr).sum()
print(f"2. Re-runing the Incorrect instances (Logistic Regression): {incorrect_instances_lr}")

X = data.drop('gender', axis=1)
y = data['gender']
lr_model.fit(X, y)

coefficients = abs(lr_model.named_steps['classifier'].coef_[0])

top2_indices = coefficients.argsort()[-2:][::-1]
top2_features = X.columns[top2_indices]
```

```
print(f"The 2 powerful attributes: {top2_features}")



X = data.drop('gender', axis=1)
y = data['gender']
top2_attributes = ['beard', 'shoe_size']



X_excluded = X.drop(columns=top2_attributes)

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X_excluded.select_dtypes(include=['float64', 'int64']).columns),
        ('cat', OneHotEncoder(), X_excluded.select_dtypes(include=['object']).columns)
    ]
)

lr_model_excluded = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])



X_train_excluded, X_test_excluded, y_train, y_test = train_test_split(X_excluded, y, test_size=0.2,
random_state=42)



lr_model_excluded.fit(X_train_excluded, y_train)
y_pred_lr_excluded = lr_model_excluded.predict(X_test_excluded)
```

incorrect_instances_lr_excluded = (y_test != y_pred_lr_excluded).sum()

print(f"Incorrect instances (Logistic Regression) after excluding the 2 powerful attributes: {incorrect_instances_lr_excluded}")

```
1. Incorrect instances (Logistic Regression): 0
2. Re-runing the Incorrect instances (Logistic Regression): 0
The 2 powerful attributes: Index(['beard', 'shoe_size'], dtype='object')
Incorrect instances (Logistic Regression) after excluding the 2 powerful attributes: 2
```