Copy_of_Lab_8_Data_Pre_processing_Homework

November 4, 2024

[ ]:
```
#Run this command before importing the dataset.
!pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-
packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in
/usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2024.8.30)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.0.0->ucimlrepo) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.0.0->ucimlrepo) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrepo) (1.16.0)
```

Load the dataset from the repository https://archive.ics.uci.edu/dataset/2/adult (4 marks)

[ ]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
adult = fetch_ucirepo(id=2)

# data (as pandas dataframes)
X = adult.data.features
y = adult.data.targets
```

[ ]:
```
X.head()
```

[ ]:
```
   age         workclass  fnlwgt  education  education-num  \
0   39         State-gov   77516  Bachelors             13
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
```

```
4   28           Private  338409  Bachelors             13

       marital-status         occupation   relationship   race     sex  \
0       Never-married       Adm-clerical  Not-in-family  White    Male
1  Married-civ-spouse    Exec-managerial        Husband  White    Male
2            Divorced  Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse     Prof-specialty           Wife  Black  Female

   capital-gain  capital-loss  hours-per-week native-country
0          2174             0              40  United-States
1             0             0              13  United-States
2             0             0              40  United-States
3             0             0              40  United-States
4             0             0              40           Cuba
```

[ ]:
```
y.head()
```

[ ]:
```
  income
0  <=50K
1  <=50K
2  <=50K
3  <=50K
4  <=50K
```

[ ]:
```
X.shape, y.shape
```

[ ]:
```
((48842, 14), (48842, 1))
```

[ ]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             48842 non-null  int64
 1   workclass       47879 non-null  object
 2   fnlwgt          48842 non-null  int64
 3   education       48842 non-null  object
 4   education-num   48842 non-null  int64
 5   marital-status  48842 non-null  object
 6   occupation      47876 non-null  object
 7   relationship    48842 non-null  object
 8   race            48842 non-null  object
 9   sex             48842 non-null  object
 10  capital-gain    48842 non-null  int64
 11  capital-loss    48842 non-null  int64
```

```
 12  hours-per-week  48842 non-null  int64
 13  native-country  48568 non-null  object
dtypes: int64(6), object(8)
memory usage: 5.2+ MB
```

[ ]:
```
X.describe()
```

[ ]:
```
                age        fnlwgt  education-num  capital-gain  capital-loss  \
count  48842.000000  4.884200e+04   48842.000000  48842.000000  48842.000000
mean      38.643585  1.896641e+05      10.078089   1079.067626     87.502314
std       13.710510  1.056040e+05       2.570973   7452.019058    403.004552
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.175505e+05       9.000000      0.000000      0.000000
50%       37.000000  1.781445e+05      10.000000      0.000000      0.000000
75%       48.000000  2.376420e+05      12.000000      0.000000      0.000000
max       90.000000  1.490400e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    48842.000000
mean        40.422382
std         12.391444
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```

[ ]:
```
X.isna().sum()
```

[ ]:
```
age                 0
workclass         963
fnlwgt              0
education           0
education-num       0
marital-status      0
occupation        966
relationship        0
race                0
sex                 0
capital-gain        0
capital-loss        0
hours-per-week      0
native-country    274
dtype: int64
```

[ ]:
```
y.isna().sum()
```

[ ]:
```
income    0
dtype: int64
```

[ ]:
```python
for cat in X.select_dtypes(include='object'):
  print(cat, len(X[cat].unique()), X[cat].unique())
```

```
workclass 10 ['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov'
'?'
 'Self-emp-inc' 'Without-pay' 'Never-worked' nan]
education 16 ['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college'
'Assoc-acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']
marital-status 7 ['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-
spouse-absent'
 'Separated' 'Married-AF-spouse' 'Widowed']
occupation 16 ['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-
specialty'
 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' '?'
 'Protective-serv' 'Armed-Forces' 'Priv-house-serv' nan]
relationship 6 ['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-
relative']
race 5 ['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']
sex 2 ['Male' 'Female']
native-country 43 ['United-States' 'Cuba' 'Jamaica' 'India' '?' 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands' nan]
```

[ ]:
```python
y.value_counts()
```

[ ]:
```
income
<=50K     24720
<=50K.    12435
>50K       7841
>50K.      3846
Name: count, dtype: int64
```

[ ]:
```python
y = y.income.str.replace('<=50K.', '<=50K')
y = y.str.replace('>50K.', '>50K')
y.value_counts()
```

[ ]:
```
income
<=50K    37155
```

```
>50K     11687
Name: count, dtype: int64
```

Split the dataset into Train and Test Dataset in 80:20 ratio (1 marks)

[ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=52)
```

[ ]:
```
for cat in X_train.select_dtypes(include='object'):
  print(cat, len(X_train[cat].unique()), X_train[cat].unique())
  print(len(X_test[cat].unique()), X_test[cat].unique())
```

```
workclass 10 ['?' 'Self-emp-not-inc' 'Private' 'Self-emp-inc' 'Federal-gov'
'Local-gov'
 'State-gov' nan 'Never-worked' 'Without-pay']
10 ['Self-emp-not-inc' 'Self-emp-inc' 'Private' 'State-gov' 'Local-gov'
 'Federal-gov' '?' nan 'Without-pay' 'Never-worked']
education 16 ['5th-6th' 'Assoc-voc' 'Assoc-acdm' 'HS-grad' 'Some-college' '9th'
 'Bachelors' 'Doctorate' '7th-8th' 'Masters' '12th' '10th' '11th'
 'Prof-school' '1st-4th' 'Preschool']
16 ['Masters' 'Some-college' 'Assoc-voc' '11th' '9th' '12th' 'HS-grad'
 'Assoc-acdm' '10th' 'Prof-school' '7th-8th' 'Bachelors' 'Doctorate'
 'Preschool' '1st-4th' '5th-6th']
marital-status 7 ['Never-married' 'Married-civ-spouse' 'Divorced' 'Widowed'
 'Married-spouse-absent' 'Separated' 'Married-AF-spouse']
7 ['Married-civ-spouse' 'Never-married' 'Divorced' 'Separated' 'Widowed'
 'Married-spouse-absent' 'Married-AF-spouse']
occupation 16 ['?' 'Farming-fishing' 'Craft-repair' 'Tech-support' 'Sales'
 'Adm-clerical' 'Exec-managerial' 'Other-service' 'Prof-specialty'
 'Machine-op-inspct' 'Transport-moving' 'Priv-house-serv' nan
 'Protective-serv' 'Handlers-cleaners' 'Armed-Forces']
16 ['Sales' 'Exec-managerial' 'Adm-clerical' 'Handlers-cleaners'
 'Craft-repair' 'Prof-specialty' 'Other-service' 'Tech-support'
 'Machine-op-inspct' 'Farming-fishing' 'Transport-moving'
 'Protective-serv' '?' nan 'Priv-house-serv' 'Armed-Forces']
relationship 6 ['Unmarried' 'Husband' 'Not-in-family' 'Wife' 'Other-relative'
'Own-child']
6 ['Husband' 'Own-child' 'Wife' 'Unmarried' 'Not-in-family' 'Other-relative']
race 5 ['White' 'Black' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']
5 ['Asian-Pac-Islander' 'White' 'Amer-Indian-Eskimo' 'Black' 'Other']
sex 2 ['Female' 'Male']
2 ['Male' 'Female']
native-country 43 ['El-Salvador' 'United-States' 'Canada' 'Mexico' 'Cuba' '?'
'Italy'
 'Philippines' 'Guatemala' 'Puerto-Rico' 'Cambodia' 'Taiwan' 'Vietnam'
 'Dominican-Republic' 'China' 'India' 'Ireland' nan 'Columbia' 'England'
```

```
 'Japan' 'South' 'Haiti' 'Laos' 'Jamaica' 'Peru' 'Germany' 'Poland'
 'Yugoslavia' 'Hong' 'Greece' 'Outlying-US(Guam-USVI-etc)' 'Scotland'
 'France' 'Iran' 'Portugal' 'Nicaragua' 'Thailand' 'Ecuador' 'Honduras'
 'Hungary' 'Trinadad&Tobago' 'Holand-Netherlands']
42 ['India' 'United-States' nan 'China' 'Haiti' 'Germany' 'Mexico'
 'Puerto-Rico' 'Cambodia' 'El-Salvador' 'Philippines' 'Columbia' 'Japan'
 'Honduras' '?' 'Dominican-Republic' 'Portugal' 'Ecuador' 'Cuba' 'Greece'
 'Vietnam' 'South' 'Guatemala' 'Hungary' 'Taiwan' 'France' 'Yugoslavia'
 'Iran' 'Italy' 'Canada' 'England' 'Nicaragua' 'Poland' 'Peru'
 'Trinadad&Tobago' 'Jamaica' 'Ireland' 'Scotland' 'Hong' 'Thailand'
 'Outlying-US(Guam-USVI-etc)' 'Laos']
```

[ ]:
```
X_train.shape, X_test.shape
```

[ ]:
```
((39073, 14), (9769, 14))
```

Create a data pipeline that does:

1. Imputation (5 marks)

2. Standardization and Scaling (5 marks)

3. Discretization (5 marks)

4. Encoding (5 marks)

5. Prediction (5 marks)

(Total 25 marks)

[ ]:
```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder,
 ↪OrdinalEncoder, MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

[ ]:
```python
age_transformer = Pipeline([
    ('scaler', StandardScaler())
])

workclass_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])

fnlwgt_transformer = Pipeline([
    ('scaler', MinMaxScaler())
])

education_transformer = Pipeline([
    ('encoder', OrdinalEncoder())
])

education_num_transformer = Pipeline([
    ('scaler', StandardScaler())
])

marital_status_transformer = Pipeline([
    ('encoder', OrdinalEncoder())
])

occupation_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OrdinalEncoder())
])

relationship_transformer = Pipeline([
    ('encoder', OrdinalEncoder())
])

race_transformer = Pipeline([
    ('encoder', OrdinalEncoder())
])

sex_transformer = Pipeline([
    ('encoder', OrdinalEncoder())
])

capital_gain_transformer = Pipeline([
    ('scaler', StandardScaler())
])

capital_loss_transformer = Pipeline([
```

```
    ('scaler', StandardScaler())
])

hours_per_week_transformer = Pipeline([
    ('scaler', StandardScaler())
])

native_country_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])


preprocessor = ColumnTransformer(
    transformers=[
        ('age_transformer', age_transformer, ['age']),
        ('workclass_transformer', workclass_transformer, ['workclass']),
        ('fnlwgt_transformer', fnlwgt_transformer, ['fnlwgt']),
        ('education_transformer', education_transformer, ['education']),
        ('education_num_transformer', education_num_transformer,␣
 ↪['education-num']),
        ('marital_status_transformer', marital_status_transformer,␣
 ↪['marital-status']),
        ('occupation_transformer', occupation_transformer, ['occupation']),
        ('relationship_transformer', relationship_transformer,␣
 ↪['relationship']),
        ('race_transformer', race_transformer, ['race']),
        ('sex_transformer', sex_transformer, ['sex']),
        ('capital_gain_transformer', capital_gain_transformer,␣
 ↪['capital-gain']),
        ('capital_loss_transformer', capital_loss_transformer,␣
 ↪['capital-loss']),
        ('hours_per_week_transformer', hours_per_week_transformer,␣
 ↪['hours-per-week']),
        ('native_country_transformer', native_country_transformer,␣
 ↪['native-country']),
    ],remainder='passthrough',)
```

[ ]:

```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)

y_train, y_test
```

[ ]: (array([0, 0, 0, …, 0, 0, 0]), array([0, 1, 0, …, 0, 0, 0]))

[ ]:
```python
#capture all categories
preprocessor.fit(X)

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

param_grid = [
    {
        'classifier': [LogisticRegression(max_iter=500)],
        'classifier__C': [0.1, 1, 10]
    },
    {
        'classifier': [SVC()],
        'classifier__C': [0.1, 1, 10],
        'classifier__kernel': ['linear', 'rbf']
    },
    {
        'classifier': [RandomForestClassifier()],
        'classifier__n_estimators': [100, 200],
        'classifier__max_depth': [10, 20]
    }
]

grid_search = GridSearchCV(model_pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_

y_pred = best_model.predict(X_test)

accuracy_score(y_test, y_pred), best_model
```

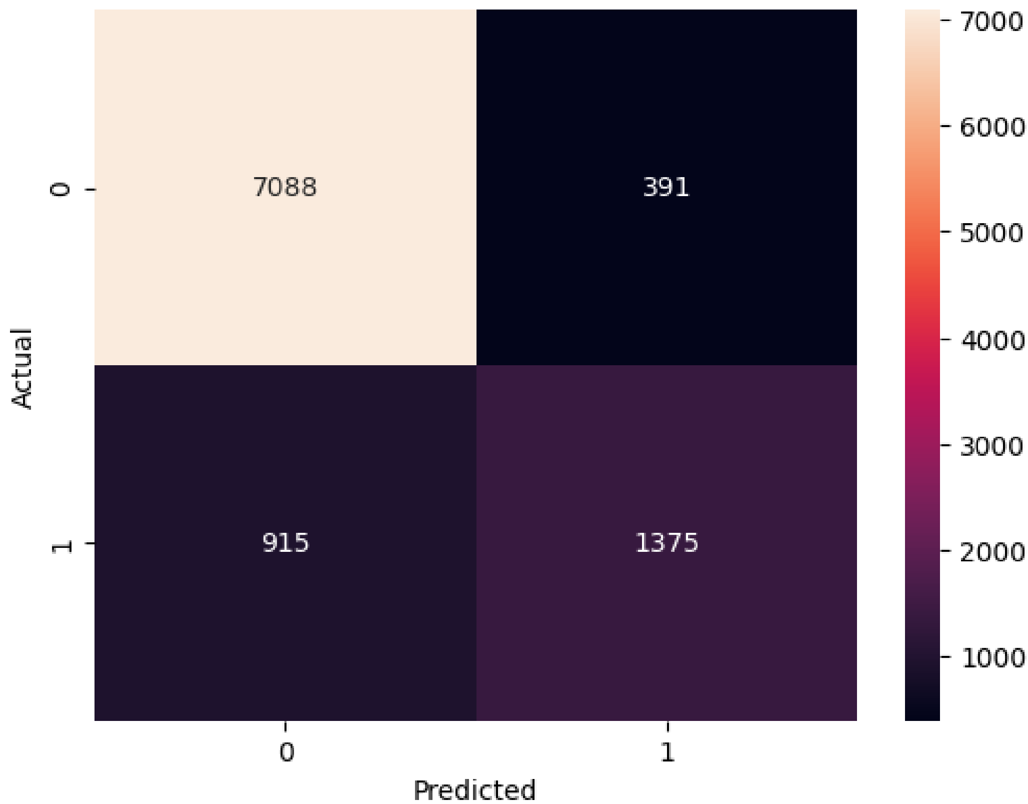
[ ]:
```
(0.8663118026410073,
 Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('age_transformer',
                                                   Pipeline(steps=[('scaler',
StandardScaler())]),
                                                   ['age']),
                                                  ('workclass_transformer',
                                                   Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                   ('encoder',
OneHotEncoder(handle_unknown='ignore'))]),
                                                   ['workclass']),
```

```
                                                  ('fnlwgt_transformer',
                                                   Pipel…
                                                   ['capital-loss']),
                                                  ('hours_per_week_transformer',
                                                   Pipeline(steps=[('scaler',
StandardScaler())]),
                                                   ['hours-per-week']),
                                                  ('native_country_transformer',
                                                   Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                   ('encoder',
OneHotEncoder(handle_unknown='ignore'))]),
                                                   ['native-country'])])),
                 ('classifier',
                  RandomForestClassifier(max_depth=20, n_estimators=200))]))
```

[ ]:

```python
print(classification_report(y_test, y_pred))
# print("Accuracy: ", accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      7479
           1       0.78      0.60      0.68      2290

    accuracy                           0.87      9769
   macro avg       0.83      0.77      0.80      9769
weighted avg       0.86      0.87      0.86      9769

[[7088  391]
 [ 915 1375]]
```


7088
391
915
1375
0
1
Actual
0
1
Predicted
7000
6000
5000
4000
3000
2000
1000