10_categorical_features

February 28, 2024

1 Building a Machine Learning workflow

Lesson 10 from Introduction to Machine Learning with scikit-learn

Note: This notebook uses Python 3.9.1 and scikit-learn 0.23.2. The original notebook (shown in the video) used Python 3.7 and scikit-learn 0.20.2.

1.1 Agenda

- Why should you use a Pipeline?
- How do you encode categorical features with OneHotEncoder?
- How do you apply OneHotEncoder to selected columns with ColumnTransformer?
- How do you build and cross-validate a Pipeline?
- How do you make predictions on new data using a Pipeline?
- Why should you use scikit-learn (rather than pandas) for preprocessing?

1.2 Step 1: Load the dataset

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv('http://bit.ly/kaggletrain')
```

[3]:
```
df.shape
```

[3]:
```
(891, 12)
```

1.3 Step 2: Select features

[4]:
```
df.columns
```

[4]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

[5]:
```
df.isna().sum()
```

[5]:
```
PassengerId      0
Survived         0
Pclass           0
```

```
Name            0
Sex             0
Age           177
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin         687
Embarked        2
dtype: int64
```

[6]:
```
df = df.loc[df.Embarked.notna(), ['Survived', 'Pclass', 'Sex', 'Embarked']]
```

[7]:
```
df.shape
```

[7]:
```
(889, 4)
```

[8]:
```
df.isna().sum()
```

[8]:
```
Survived    0
Pclass      0
Sex         0
Embarked    0
dtype: int64
```

[9]:
```
df.head()
```

[9]:

	Survived	Pclass	Sex	Embarked
0	0	3	male	S
1	1	1	female	C
2	1	3	female	S
3	1	1	female	S
4	0	3	male	S

1.4 Step 3: Cross-validate a model with one feature

[10]:
```
X = df.loc[:, ['Pclass']]
y = df.Survived
```

[11]:
```
X.shape
```

[11]:
```
(889, 1)
```

[12]:
```
y.shape
```

[12]:
```
(889,)
```

[13]:
```
from sklearn.linear_model import LogisticRegression
```

[14]:
```
logreg = LogisticRegression()
```

[15]:
```
from sklearn.model_selection import cross_val_score
```

[16]:
```
cross_val_score(logreg, X, y, cv=5, scoring='accuracy').mean()
```

[16]:
```
0.6783406335301212
```

[17]:
```
y.value_counts(normalize=True)
```

[17]:
```
0    0.617548
1    0.382452
Name: Survived, dtype: float64
```

1.5 Step 4: Encode categorical features

[18]:
```
df.head()
```

[18]:

	Survived	Pclass	Sex	Embarked
0	0	3	male	S
1	1	1	female	C
2	1	3	female	S
3	1	1	female	S
4	0	3	male	S

[19]:
```
# dummy encoding of categorical features
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(sparse=False)
```

[20]:
```
ohe.fit_transform(df[['Sex']])
```

[20]:
```
array([[0., 1.],
       [1., 0.],
       [1., 0.],
       …,
       [1., 0.],
       [0., 1.],
       [0., 1.]])
```

[21]:
```
ohe.categories_
```

[21]:
```
[array(['female', 'male'], dtype=object)]
```

[22]:
```
ohe.fit_transform(df[['Embarked']])
```

[22]:
```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
```

```
       …,
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

[23]:
```
ohe.categories_
```

[23]:
```
[array(['C', 'Q', 'S'], dtype=object)]
```

1.6 Step 5: Cross-validate a Pipeline with all features

[24]:
```
X = df.drop('Survived', axis='columns')
```

[25]:
```
X.head()
```

[25]:

	Pclass	Sex	Embarked
0	3	male	S
1	1	female	C
2	3	female	S
3	1	female	S
4	3	male	S

[26]:
```
# use when different features need different preprocessing
from sklearn.compose import make_column_transformer
```

[27]:
```
column_trans = make_column_transformer(
    (OneHotEncoder(), ['Sex', 'Embarked']),
    remainder='passthrough')
```

[28]:
```
column_trans.fit_transform(X)
```

[28]:
```
array([[0., 1., 0., 0., 1., 3.],
       [1., 0., 1., 0., 0., 1.],
       [1., 0., 0., 0., 1., 3.],
       …,
       [1., 0., 0., 0., 1., 3.],
       [0., 1., 1., 0., 0., 1.],
       [0., 1., 0., 1., 0., 3.]])
```

[29]:
```
# chain sequential steps together
from sklearn.pipeline import make_pipeline
```

[30]:
```
pipe = make_pipeline(column_trans, logreg)
```

[31]:
```
# cross-validate the entire process
# thus, preprocessing occurs within each fold of cross-validation
cross_val_score(pipe, X, y, cv=5, scoring='accuracy').mean()
```

[31]:
```
0.7727924839713071
```

1.7 Step 6: Make predictions on "new" data

[32]:
```
# added empty cell so that the cell numbering matches the video
```

[33]:
```
X_new = X.sample(5, random_state=99)
X_new
```

[33]:

	Pclass	Sex	Embarked
599	1	male	C
512	1	male	S
273	1	male	C
215	1	female	C
790	3	male	Q

[34]:
```
pipe.fit(X, y)
```

[34]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(),
                                                  ['Sex', 'Embarked'])])),
                ('logisticregression', LogisticRegression())])
```

[35]:
```
pipe.predict(X_new)
```

[35]:
```
array([1, 0, 1, 1, 0])
```

1.8 Recap

[36]:
```
import pandas as pd
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
```

[37]:
```
df = pd.read_csv('http://bit.ly/kaggletrain')
df = df.loc[df.Embarked.notna(), ['Survived', 'Pclass', 'Sex', 'Embarked']]
X = df.drop('Survived', axis='columns')
y = df.Survived
```

[38]:
```
column_trans = make_column_transformer(
    (OneHotEncoder(), ['Sex', 'Embarked']),
    remainder='passthrough')
logreg = LogisticRegression(solver='lbfgs')
```

[39]:
```
pipe = make_pipeline(column_trans, logreg)
```

[40]:
```
cross_val_score(pipe, X, y, cv=5, scoring='accuracy').mean()
```

[40]:
```
0.7727924839713071
```

[41]:
```
X_new = X.sample(5, random_state=99)
```

[42]:
```
pipe.fit(X, y)
pipe.predict(X_new)
```

[42]:
```
array([1, 0, 1, 1, 0])
```

1.9 Comments or Questions?

- Email: kevin@dataschool.io
- Website: https://www.dataschool.io
- Twitter: @justmarkham