Branch: master ▾ Car-Evaluation / car.ipynb Find file Copy path

sin121212 Add files via upload bfa6e0c on Sep 6

1 contributor

944 lines (943 sloc) 27.7 KB Raw Blame History

1. Data Preparation

1.1 Import Car Evaluation Dataset

In [1]:
```
import pandas as pd
# Import dataset
car_df = pd.read_csv(r'C:\Users\user\Desktop\Data_Science\Car\car_evaluation.csv')
# first 5 rows
car_df.head(3)
```

Out[1]:

	buying_price	maintenance_cost	number_of_doors	number_of_persons	lug_boot	safety	decision
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc

1.2 Check Data Structure

There are 1728 rows and 7 category variables in dataset.

In [2]:
```
print("column name:")
print(car_df.columns)
print("row & column:")
print(car_df.shape)
print("data type:")
print(car_df.dtypes)
```

```
column name:
Index(['buying_price', 'maintenance_cost', 'number_of_doors',
       'number_of_persons', 'lug_boot', 'safety', 'decision'],
      dtype='object')
row & column:
(1728, 7)
```

```
data type:
buying_price         object
maintenance_cost     object
number_of_doors      object
number_of_persons    object
lug_boot             object
safety               object
decision             object
dtype: object
```

1.3 OneHot Encoding Processing

Convert category dependent variable to dummy variable.

In [3]:
```
# Select all category variables exclude dependent variable "decision"
car_dummy = pd.get_dummies(car_df,drop_first=True,columns=car_df.columns.drop('decision'))
car_dummy.head(3)
```

Out[3]:

	decision	buying_price_low	buying_price_med	buying_price_vhigh	maintenance_cost_low	maintenance_cost_
0	unacc	0	0	1	0	0
1	unacc	0	0	1	0	0
2	unacc	0	0	1	0	0

2. Define Variable in Model

2.1 Independent Variable (X)

15 features in model

In [4]:
```
X = car_dummy.drop('decision', axis=1)

Number_Feature = X.shape[1]
print(Number_Feature)
# Show first 3 rows only
```

```
# Show first 3 rows only
X.head(3)
```

15

Out[4]:

	buying_price_low	buying_price_med	buying_price_vhigh	maintenance_cost_low	maintenance_cost_med	mai
0	0	0	1	0	0	1
1	0	0	1	0	0	1
2	0	0	1	0	0	1

2.2 Dependent Variable (Y)

In [5]:
```
# Define dependent variable in model
Y = car_dummy['decision']
# Show first 3 rows only
Y.head(3)
```

Out[5]:
```
0    unacc
1    unacc
2    unacc
Name: decision, dtype: object
```

3. Generate Classification Model

3.1 Model Comparison

Compare 5 classification model accuracy. To aviod overfitting problem, 5-folds cross-vaildation has applied.

In [6]:
```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
```

```
# Prepare models
models = []
# Multi-class logistic regression
models.append(('LogisticRegression', LogisticRegression(solver='lbfgs',multi_class='multinomial')))
models.append(('LinearDiscriminantAnalysis', LinearDiscriminantAnalysis()))
models.append(('KNeighbors', KNeighborsClassifier()))
models.append(('DecisionTree', DecisionTreeClassifier()))
models.append(('GaussianNB', GaussianNB()))

# Evaluate each model in turn
results = []
names = []

# Generate series of classification models
for name, model in models:
    # Repeated K-fold croos-vaildation
    kfold = model_selection.RepeatedKFold(n_splits=5, n_repeats=2, random_state=123)
    # Choose the best cross-vaildation result within each model
    cv_results = model_selection.cross_val_score(model, X, Y, cv=kfold, scoring='accuracy')
    # Store average accuracy among K-fold croos-vaildation
    results.append(cv_results.mean())
    names.append(name)

# Store model accuracy of classification methods into dataframe
list_of_tuples = list(zip(names, results))
compare_model_df = pd.DataFrame(list_of_tuples,columns = ['Model', 'Accuracy'])
# Sort by accuracy
compare_model_df = compare_model_df.sort_values(by=['Accuracy'],ascending=False)

compare_model_df
```

Out[6]:

	Model	Accuracy
3	DecisionTree	0.915217
0	LogisticRegression	0.900751
1	LinearDiscriminantAnalysis	0.888892
2	KNeighbors	0.817133
4	GaussianNB	0.489004

3.2 Turning Decision Tree Model Parameters with Grid Search

Although decision tree model has the highest accuracy (91.46%), the model performance can be improved by turning parameters.

In [7]:
```
import numpy as np
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth': np.arange(1, 20)}

tree = GridSearchCV(DecisionTreeClassifier(), param_grid,cv=kfold)

tree.fit(X,Y)
```

Out[7]:
```
GridSearchCV(cv=<sklearn.model_selection._split.RepeatedKFold object at 0x00000272F94DD7B8>,
             error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=None,
                                              splitter='best'),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
       15, 16, 17,
              18, 19])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

3.3 Best Model Parameters in Grid Search Result

In [8]:
```
tree.best_estimator_
```

Out[8]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=10

```
Out[8]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=10,
                               max_features=None, max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, presort=False,
                               random_state=None, splitter='best')
```

3.4 Classification Result -- Confusion Matrix

In [9]:
```
CM = pd.crosstab(Y, tree.predict(X),rownames=['Actual'], colnames=['Predicted'])
CM
```

Out[9]:

Predicted	acc	good	unacc	vgood
Actual				
acc	374	5	1	4
good	8	58	0	3
unacc	24	2	1184	0
vgood	0	2	0	63