2020-2021年冬季学期数据分析与智能计算

综合应用题（共1题，34分）
台风记录数据集（winds-cm.csv）记录了2014年某区域发生的台风信息，包括台风名、台风等级、气压（百帕）、移动速度（公里/时)、纬度、经度、记录数、顺序、风速（米/秒）等9个属性(具体说明见“数据集说明”文件)。试分析与台风等级相关的特征，并建立等级判别模型。

具体要求如下：

1)从文件中读出台风数据（3分）；
2)数据集中表示台风等级level有六个等级为：热带低压、热带风暴、强热带风暴、台风、强台风、超强台风。将台风等级字符串依次替换为数字1-6（4分）；
3)计算台风的各个特征与台风等级的相关性，筛选出相关性较高（相关系数>0.6）的特征建立数据集（5分）；
4)绘制图形展示筛选出的特征与台风等级的相关性（4分）；
5)按照合适比例将分析数据分为训练集和测试集（3分）；
6)在训练集上建立决策树分类模型（7分）；
7)在测试集上测试分类模型的性能（3分）；
8) 根据第7）步的运行结果，说明分类模型在台风等级判别上的性能（5分）。

自己考前随便写的答案 仅供参考

2020-2021年冬季学期数据分析与智能计算

In [11]:
```
#导入库文件
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [12]:
```
#1 从文件中读出台风数据;
filename='winds-cm.csv'
data=pd.read_csv(filename,index_col=False)
data
```

Out[12]:

	windname	level	pressure	movespeed	latitude	longtitude	count	number	windspeed
0	威马逊（Rammasun）	热带低压	998	20.0	23.3	104.6	1	107	16.0
1	威马逊（Rammasun）	热带风暴	996	20.0	23.0	105.1	1	106	18.0
2	威马逊（Rammasun）	热带风暴	994	17.0	22.9	105.6	1	105	20.0
3	威马逊（Rammasun）	热带风暴	990	20.0	22.9	106.2	1	104	23.0
4	威马逊（Rammasun）	热带风暴	990	20.0	22.8	106.4	1	103	23.0
...	...	...	...	...	...	...	...	...	...
388	法茜（Faxai）	热带风暴	996	8.0	9.0	150.1	1	5	20.0
389	法茜（Faxai）	热带风暴	998	6.0	8.7	149.6	1	4	18.0
390	法茜（Faxai）	热带风暴	998	5.0	8.6	149.5	1	3	18.0
391	法茜（Faxai）	热带风暴	998	5.0	9.0	149.0	1	2	18.0
392	法茜（Faxai）	热带风暴	998	5.0	8.7	148.2	1	1	18.0

393 rows × 9 columns

In [13]:
```
#2 替换代表台风等级的字符串
data.loc[data['level']=='热带低压','level']=1
data.loc[data['level']=='热带风暴','level']=2
data.loc[data['level']=='强热带风暴','level']=3
data.loc[data['level']=='台风','level']=4
data.loc[data['level']=='强台风','level']=5
data.loc[data['level']=='超强台风','level']=6
data['level']=data['level'].values.astype(int)
data
```

Out[13]:

	windname	level	pressure	movespeed	latitude	longtitude	count	number	windspeed
0	威马逊 (Rammasun)	1	998	20.0	23.3	104.6	1	107	16.0
1	威马逊 (Rammasun)	2	996	20.0	23.0	105.1	1	106	18.0
2	威马逊 (Rammasun)	2	994	17.0	22.9	105.6	1	105	20.0
3	威马逊 (Rammasun)	2	990	20.0	22.9	106.2	1	104	23.0
4	威马逊 (Rammasun)	2	990	20.0	22.8	106.4	1	103	23.0
...	...	...	...	...	...	...	...	...	...
388	法茜 (Faxai)	2	996	8.0	9.0	150.1	1	5	20.0
389	法茜 (Faxai)	2	998	6.0	8.7	149.6	1	4	18.0
390	法茜 (Faxai)	2	998	5.0	8.6	149.5	1	3	18.0
391	法茜 (Faxai)	2	998	5.0	9.0	149.0	1	2	18.0
392	法茜 (Faxai)	2	998	5.0	8.7	148.2	1	1	18.0

393 rows × 9 columns

In [14]:
```
#3 计算台风的各个特征与台风等级的相关性
data.corr()
```

Out[14]:

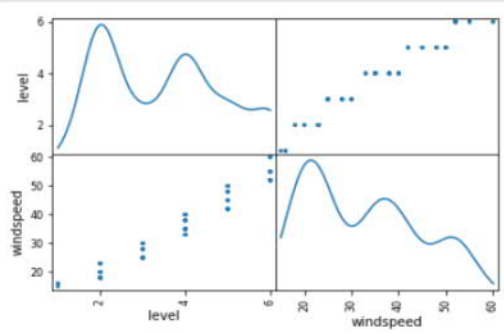
	level	pressure	movespeed	latitude	longtitude	count	number	windspeed
level	1.000000	-0.962942	0.153720	0.169610	-0.229580	NaN	0.215798	0.977935
pressure	-0.962942	1.000000	-0.151448	-0.215142	0.295134	NaN	-0.253137	-0.990008
movespeed	0.153720	-0.151448	1.000000	0.424086	-0.061812	NaN	0.298140	0.158259
latitude	0.169610	-0.215142	0.424086	1.000000	-0.358001	NaN	0.666028	0.186772
longtitude	-0.229580	0.295134	-0.061812	-0.358001	1.000000	NaN	-0.652983	-0.269387
count	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
number	0.215798	-0.253137	0.298140	0.666028	-0.652983	NaN	1.000000	0.248448
windspeed	0.977935	-0.990008	0.158259	0.186772	-0.269387	NaN	0.248448	1.000000

In [15]:
```
# 筛选出相关性较高的特征建立数据集
winds=data[['level','windspeed']]
winds.index=data['windname']
winds
```

Out[15]:

windname	level	windspeed
威马逊 (Rammasun)	1	16.0
威马逊 (Rammasun)	2	18.0
威马逊 (Rammasun)	2	20.0
威马逊 (Rammasun)	2	23.0
威马逊 (Rammasun)	2	23.0
...	...	...
法茜 (Faxai)	2	20.0
法茜 (Faxai)	2	18.0
法茜 (Faxai)	2	18.0
法茜 (Faxai)	2	18.0
法茜 (Faxai)	2	18.0

In [16]:
```
#缺失数据处理
winds=winds.dropna().copy()
#4 绘图展示相关性
pd.plotting.scatter_matrix(winds,diagonal='kde')
#kde与hist二选一
plt.show()
```



In [17]:
```
#5 按照合适比例将分析数据分为训练集和测试集
from sklearn import model_selection
from sklearn import preprocessing
X=winds.loc[:,['windspeed']].values.astype(float)
y=winds.loc[:,['level']].values.astype(int)
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.35, random_state=1)
```

In [18]:
```
#6训练模型，预测样本分类
#1 决策树
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
clf.score(X_train,y_train)

#2 线性回归
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train,y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
#7 评估分类器性能,计算R*R、混淆矩阵，Precision 和 Recall
from sklearn import metrics
y_test_predict=linreg.predict(X_test)
test_err = metrics.mean_squared_error(y_test, y_test_predict)
test_err

predicted_y_test = clf.predict(X_test)
from sklearn import metrics
print(metrics.classification_report(y_test, predicted_y_test))
print('Confusion matrix:' )
print( metrics.confusion_matrix(y_test, predicted_y_test) )
```

In [19]:
```
#7 评估分类器性能,计算R*R、混淆矩阵，Precision 和 Recall
from sklearn import metrics
y_test_predict=linreg.predict(X_test)
test_err = metrics.mean_squared_error(y_test, y_test_predict)
test_err

predicted_y_test = clf.predict(X_test)
from sklearn import metrics
print(metrics.classification_report(y_test, predicted_y_test))
print('Confusion matrix:' )
print( metrics.confusion_matrix(y_test, predicted_y_test) )
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00        49
           3       1.00      1.00      1.00        14
           4       1.00      1.00      1.00        32
           5       1.00      1.00      1.00        16
           6       1.00      1.00      1.00        19

    accuracy                           1.00       133
   macro avg       1.00      1.00      1.00       133
weighted avg       1.00      1.00      1.00       133

Confusion matrix:
[[ 3  0  0  0  0  0]
 [ 0 49  0  0  0  0]
 [ 0  0 14  0  0  0]
 [ 0  0  0 32  0  0]
 [ 0  0  0  0 16  0]
 [ 0  0  0  0  0 19]]
```

In [20]:
```
#8 虽然此模型在测试集上效果很好，但是实际情况...（自己写的东西吧）
```