1

In [104]:
```
df_train['2011 revenue'].std()
```

Out[104]: 17637.619529880183

In [105]:
```
df_train['2010 revenue'].median()
```

Out[105]: 348.08

In [106]:
```
df_train['2011 revenue'].median()
```

Out[106]: 1664.98

In [107]:
```
df_train["2010 revenue"] < 5037.47039
df_train[df_train["2010 revenue"] < 5037.47039]
df_train["2011 revenue"] < 54577.8385
df_train[df_train["2011 revenue"] < 54577.8385]
df_train.head()
```

Out[107]:

	2010 revenue	days_since_first_purchase	days_since_last_purchase	number_of_purchase	avg_order_cost	2011 revenue
1	711.79	23.0	23.0	1.0	711.79	3598.21
2	892.80	14.0	14.0	1.0	892.80	904.44
20	1868.02	16.0	13.0	2.0	934.01	1677.67
26	1001.52	10.0	10.0	1.0	1001.52	626.60
31	600.72	8.0	8.0	1.0	600.72	1249.84

2

In [108]:
```
df_train.corr()['2011 revenue']
#最相關 2010rev 最不相關= days_since_first_purchase
```

Out[108]:
```
2010 revenue                0.649649
days_since_first_purchase   0.040175
days_since_last_purchase   -0.104856
number_of_purchase          0.282602
avg_order_cost              0.623672
2011 revenue                1.000000
Name: 2011 revenue, dtype: float64
```

In [109]:
```
df_train = df_train.drop("days_since_first_purchase", axis = 1)
df_train.head()
```

Out[109]:

	2010 revenue	days_since_last_purchase	number_of_purchase	avg_order_cost	2011 revenue
1	711.79	23.0	1.0	711.79	3598.21
2	892.80	14.0	1.0	892.80	904.44
20	1868.02	13.0	2.0	934.01	1677.67
26	1001.52	10.0	1.0	1001.52	626.60
31	600.72	8.0	1.0	600.72	1249.84

3、 4、5

In [110]:
```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score




x_train, x_test, y_train, y_test = train_test_split(df_train[['2010 revenue','days_since_last_purchase','number_of_purchase'
```

In [111]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train.astype('int'))
y_pred = clf.predict(x_test)

print(model.coef_)

# 印出截距
print(model.intercept_ )

#2011rev = 5.392'2010 revenue'+41.2639895'days_since_last_purchase'+200.97977002'number_of_purchase'+6.96888101'avg_order_co
```

```
[[  5.39200819  41.2639895   200.97977002   6.96888101]]
[-2643.79605647]
```

6

```
In [112]:
predicted_rev = model.predict(x_test)
predicted_rev[:]
```

```
Out[112]: array([[ 1.05643979e+04],
                 [ 1.97076508e+03],
                 [ 8.85543568e+02],
                 [ 1.92251527e+03],
                 [ 2.37671848e+03],
                 [ 4.33736934e+03],
                 [ 1.62457975e+03],
                 [ 4.12551571e+03],
                 [ 3.13243719e+03],
                 [-3.26066118e+02],
                 [-1.00145307e+03],
                 [ 6.92915761e+03],
                 [ 4.90394478e+03],
                 [ 8.21556923e+02],
                 [ 6.25322162e+03],
                 [-3.12041167e+01],
                 [ 4.95267535e+02],
                 [ 3.12385806e+03],
                 [ 8.34755313e+03],
                 [ 1.26680819e+03],
```

```
In [113]:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 4150.979762845849
Root Mean Squared Error: 12301.60463444743
```

7

3	60.182599	86.308552	1
4	79.032736	75.344376	1

```
In [115]:
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score



x_train, x_test, y_train, y_test = train_test_split(df_train[['0','1']], df_train[['2']], test_size=0.3, random_state=1)
```

```
In [ ]:
```

```
In [116]:
from sklearn.linear_model  import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
Out[116]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, l1_ratio=None, max_iter=100,
                             multi_class='auto', n_jobs=None, penalty='l2',
                             random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                             warm_start=False)
```

```
In [122]:
from sklearn import preprocessing, linear_model
logistic_regr = linear_model.LogisticRegression()
```

8

In [116]:
```
from sklearn.linear_model  import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[116]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [122]:
```
from sklearn import preprocessing, linear_model
logistic_regr = linear_model.LogisticRegression()
logistic_regr.fit(df_train[['0','1']], df_train['2'])


predictions = lr.predict(df_train[['0','1']])
accuracy = lr.score(df_train[['0','1']], df_train['2'] )
print(accuracy)
```

```
0.91
```

In [134]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

9

```

```

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

pa = df_train[df_train["2"] == 1]
fail = df_train[df_train["2"] == 0]
plt.scatter(pa["0"], pa["1"], label="pa", color="blue", marker="o", alpha=0.5)
plt.scatter(fail["0"], fail["1"], label="fail", color="red", marker="x", alpha=0.5)
plt.xlabel("subject 0")
plt.ylabel("subject 1")
plt.title("logistic regression decision boundary")
plt.legend()
plt.show()
```

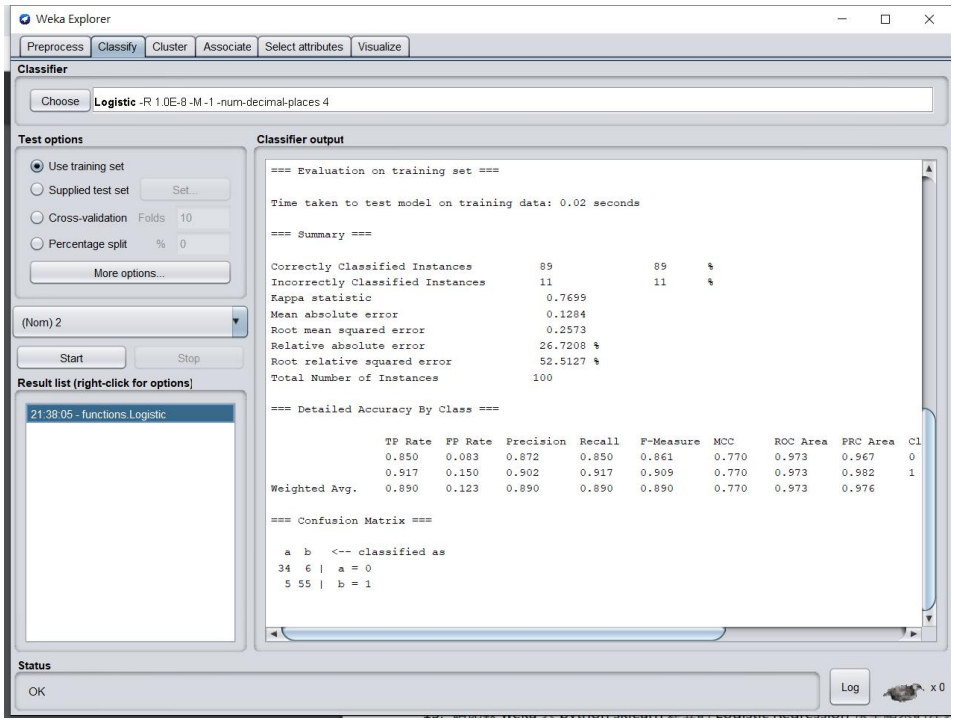
[ ]:
```

```

[ ]:
```

```

12



13

weka 因為是以 100%資料訓練模型，再以自身原始 data 做測試，如此一來的準確率會比 python 只有 70%資料訓練來的準(且因 weka 是以自身資料做測試，也許會有一點 overfitting 嫌疑)