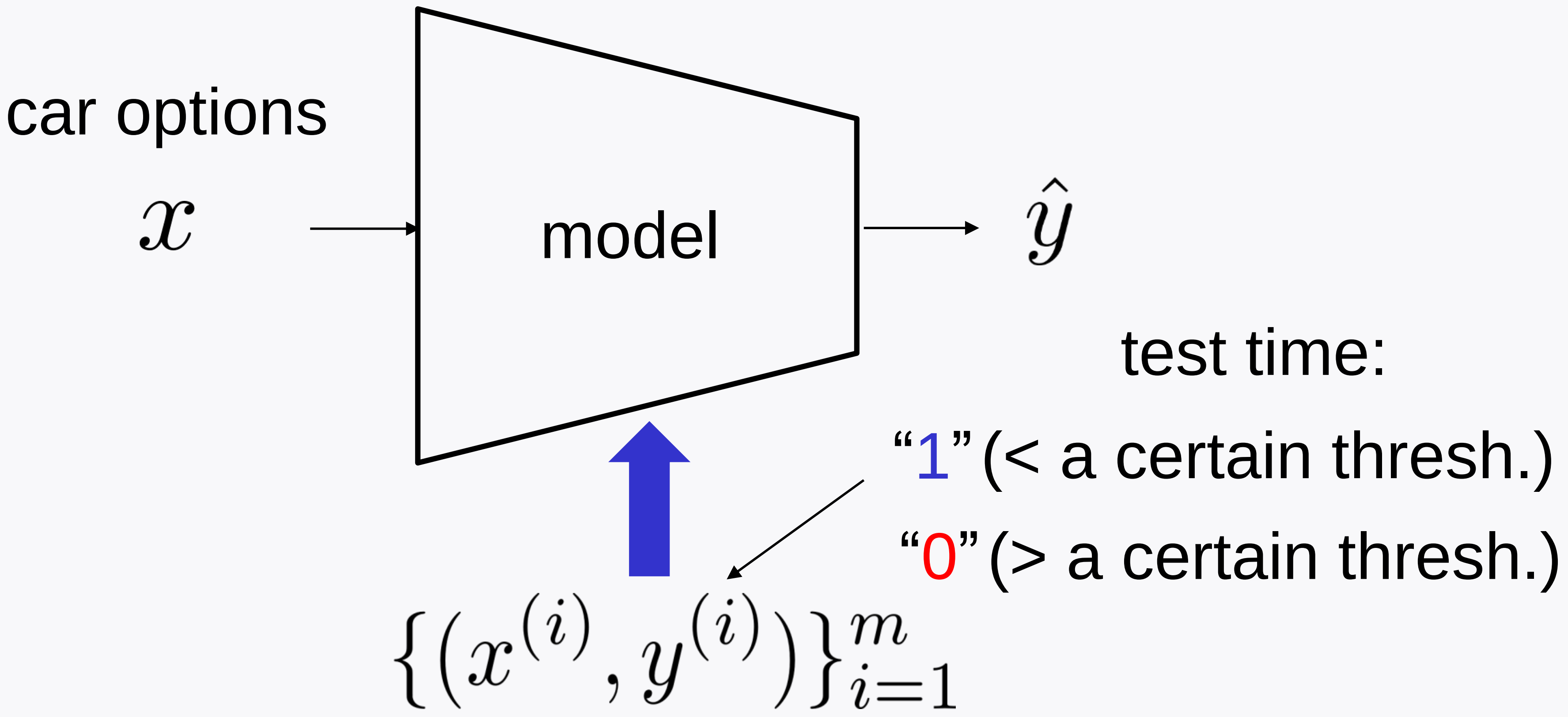
Mini-project #1

Practice Session 17

Changho Suh

January 29, 2024

Recap: Test-time prediction



car options

x

model

$\hat{y}$

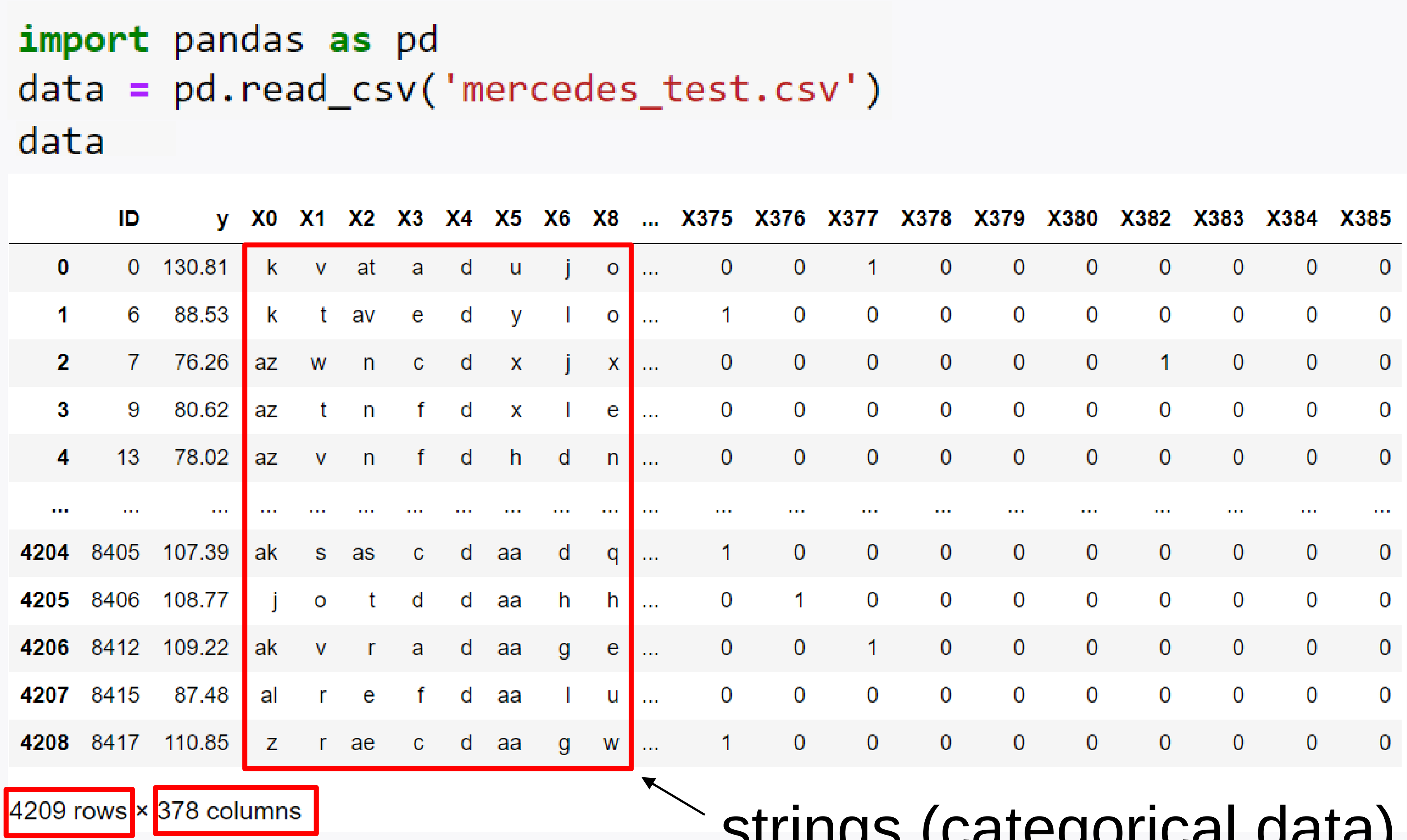
test time:

"1" (< a certain thresh.)

"0" (> a certain thresh.)

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$$

Recap: Loading MB dataset



```python
import pandas as pd
data = pd.read_csv('mercedes_test.csv')
data
```

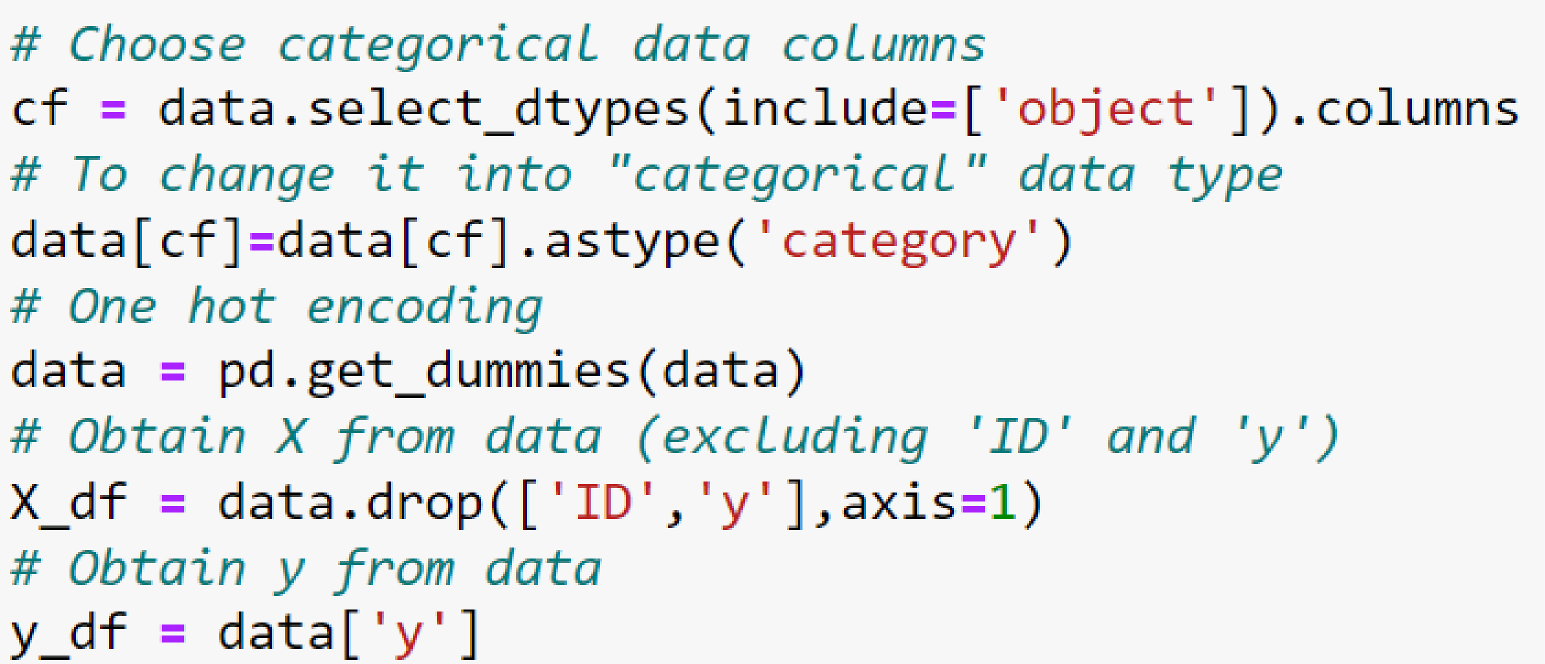
	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	0	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8405	107.39	ak	s	as	c	d	aa	d	q	...	1	0	0	0	0	0	0	0	0	0
4205	8406	108.77	j	o	t	d	d	aa	h	h	...	0	1	0	0	0	0	0	0	0	0
4206	8412	109.22	ak	v	r	a	d	aa	g	e	...	0	0	1	0	0	0	0	0	0	0
4207	8415	87.48	al	r	e	f	d	aa	l	u	...	0	0	0	0	0	0	0	0	0	0
4208	8417	110.85	z	r	ae	c	d	aa	g	w	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 378 columns

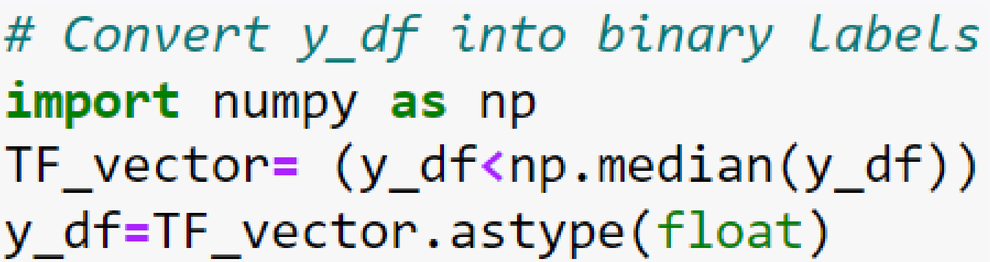
strings (categorical data)

$$m = 4209 \quad n = 376(= 378 - 2)$$

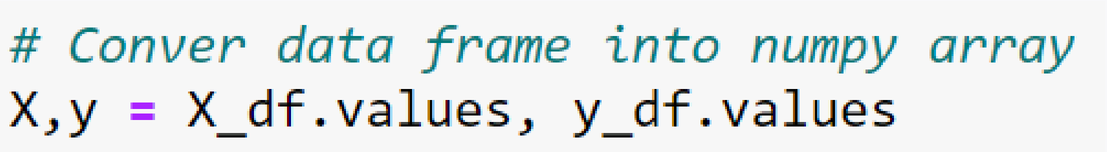
Recap: Preprocessing



```
# Choose categorical data columns
cf = data.select_dtypes(include=['object']).columns
# To change it into "categorical" data type
data[cf]=data[cf].astype('category')
# One hot encoding
data = pd.get_dummies(data)
# Obtain X from data (excluding 'ID' and 'y')
X_df = data.drop(['ID','y'],axis=1)
# Obtain y from data
y_df = data['y']
```

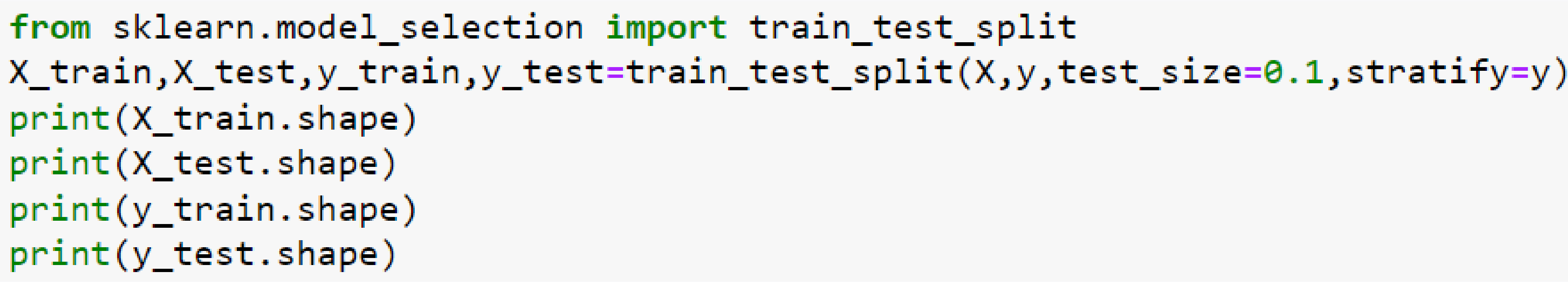


```
# Convert y_df into binary labels
import numpy as np
TF_vector= (y_df<np.median(y_df))
y_df=TF_vector.astype(float)
```

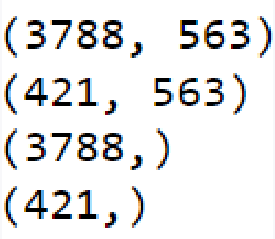


```
# Conver data frame into numpy array
X,y = X_df.values, y_df.values
```

Recap: Split into train and test datasets

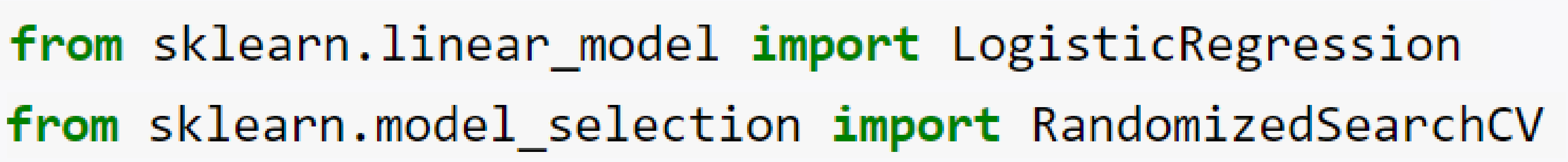


```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,stratify=y)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

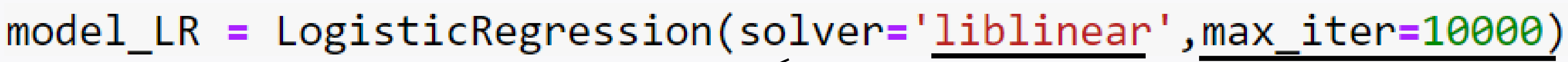


```
(3788, 563)
(421, 563)
(3788,)
(421,)
```

Model: Logistic Regression



```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
```



```
model_LR = LogisticRegression(solver='liblinear',max_iter=10000)
```

default='lbfgs'
support l2 only

algorithm name
support l1, l2
regularization

default=100

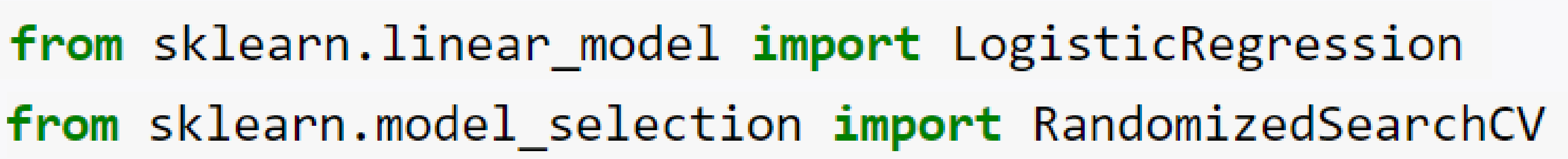
$$\min_{w} \frac{1}{m}\sum_{i=1}^{m} \ell(y^{(i)}, \hat{y}^{(i)}) + \lambda\boxed{\|w\|^2} \longleftarrow$$

l2 norm

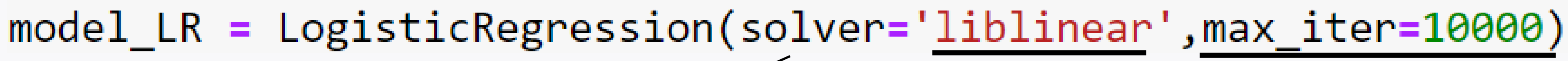
$$\|w\|_1 := |w_1| + \cdots + |w_n|$$

(l1 norm)

Model: Logistic Regression



```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
```

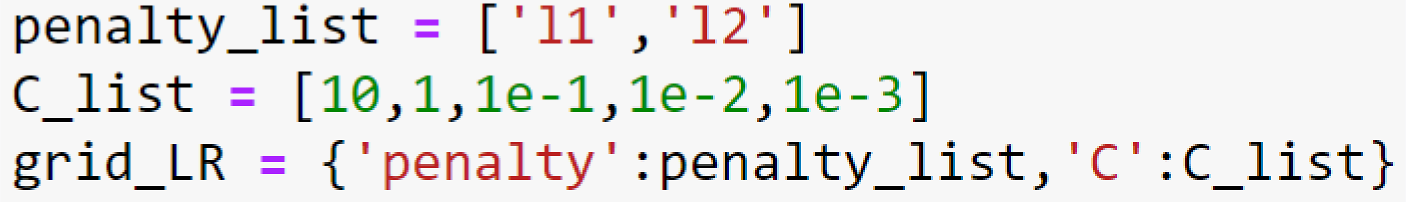


```
model_LR = LogisticRegression(solver='liblinear',max_iter=10000)
```

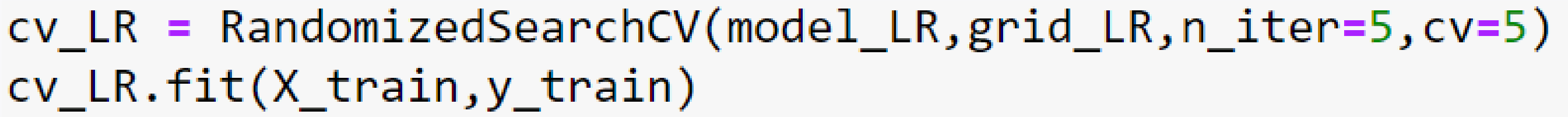
default='lbfgs'
support l2 only

algorithm name
support l1, l2
regularization

default=100

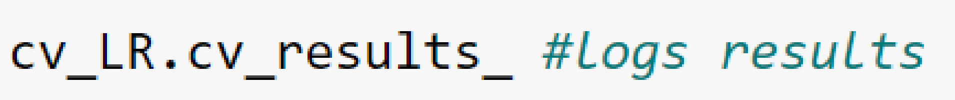


```
penalty_list = ['l1','l2']
C_list = [10,1,1e-1,1e-2,1e-3]
grid_LR = {'penalty':penalty_list,'C':C_list}
```

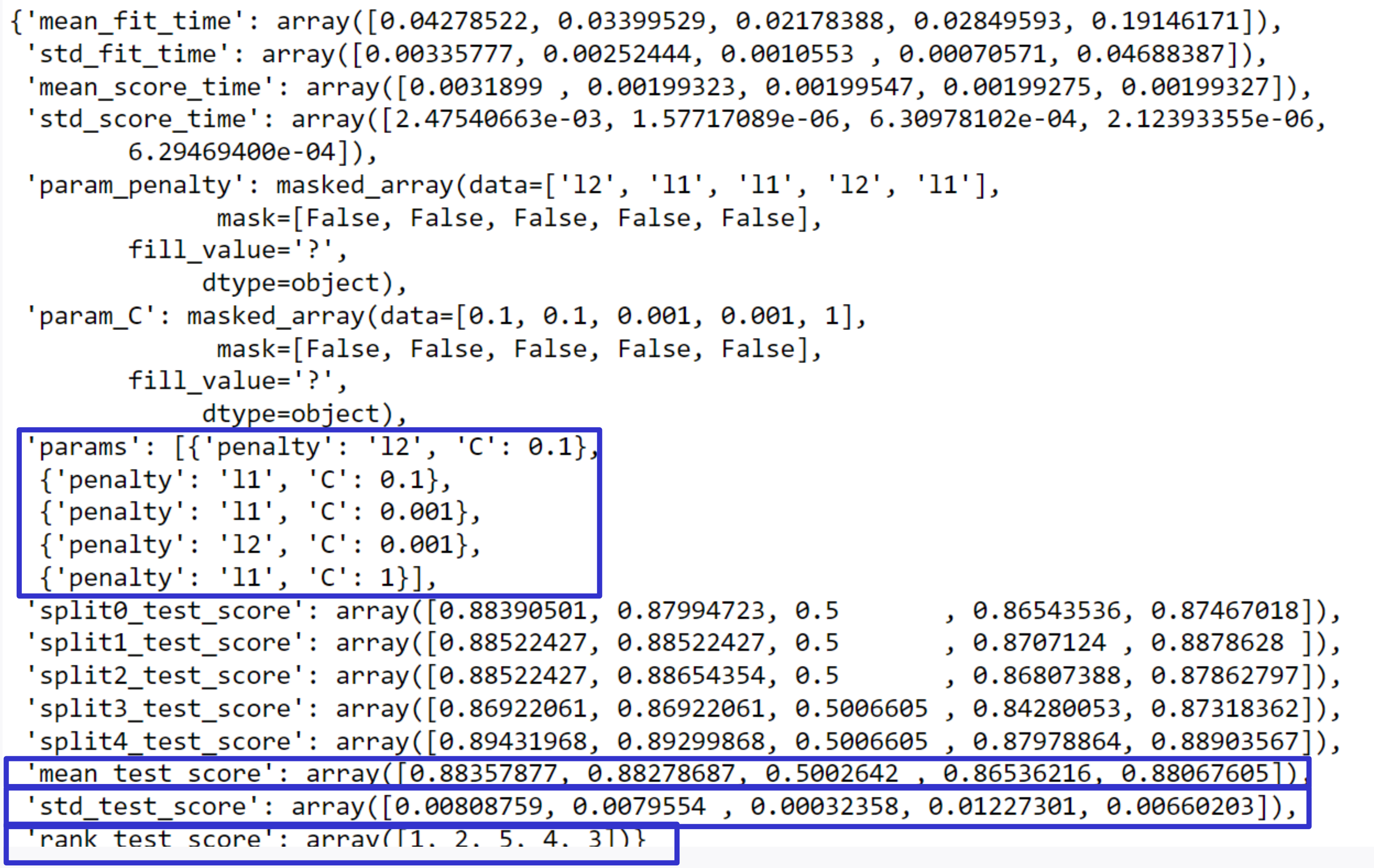


```
cv_LR = RandomizedSearchCV(model_LR,grid_LR,n_iter=5,cv=5)
cv_LR.fit(X_train,y_train)
```

Logs results



```
cv_LR.cv_results_ #logs results
```

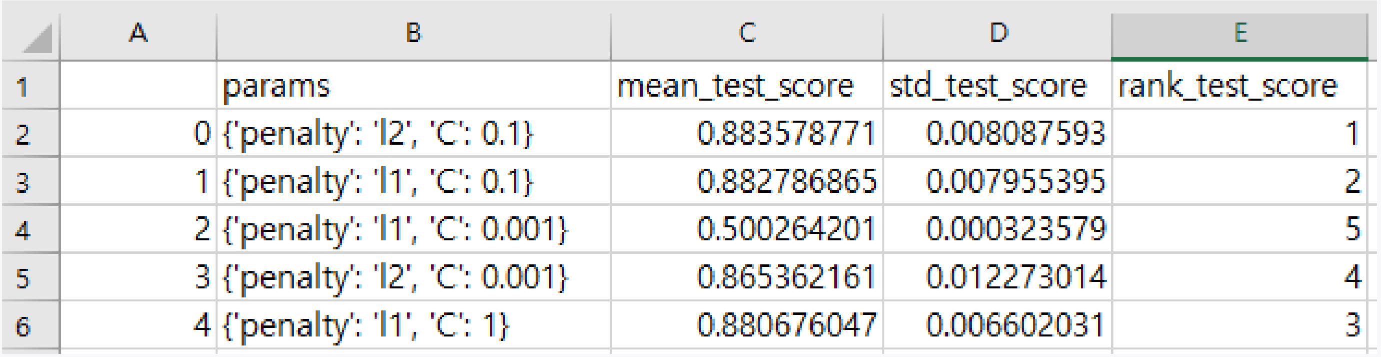


```
{'mean_fit_time': array([0.04278522, 0.03399529, 0.02178388, 0.02849593, 0.19146171]),
 'std_fit_time': array([0.00335777, 0.00252444, 0.0010553 , 0.00070571, 0.04688387]),
 'mean_score_time': array([0.0031899 , 0.00199323, 0.00199547, 0.00199275, 0.00199327]),
 'std_score_time': array([2.47540663e-03, 1.57717089e-06, 6.30978102e-04, 2.12393355e-06,
        6.29469400e-04]),
 'param_penalty': masked_array(data=['l2', 'l1', 'l1', 'l2', 'l1'],
              mask=[False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_C': masked_array(data=[0.1, 0.1, 0.001, 0.001, 1],
              mask=[False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'penalty': 'l2', 'C': 0.1},
  {'penalty': 'l1', 'C': 0.1},
  {'penalty': 'l1', 'C': 0.001},
  {'penalty': 'l2', 'C': 0.001},
  {'penalty': 'l1', 'C': 1}],
 'split0_test_score': array([0.88390501, 0.87994723, 0.5       , 0.86543536, 0.87467018]),
 'split1_test_score': array([0.88522427, 0.88522427, 0.5       , 0.8707124 , 0.8878628 ]),
 'split2_test_score': array([0.88522427, 0.88654354, 0.5       , 0.86807388, 0.87862797]),
 'split3_test_score': array([0.86922061, 0.86922061, 0.5006605 , 0.84280053, 0.87318362]),
 'split4_test_score': array([0.89431968, 0.89299868, 0.5006605 , 0.87978864, 0.88903567]),
 'mean_test_score': array([0.88357877, 0.88278687, 0.5002642 , 0.86536216, 0.88067605]),
 'std_test_score': array([0.00808759, 0.0079554 , 0.00032358, 0.01227301, 0.00660203]),
 'rank_test_score': array([1, 2, 5, 4, 3])}
```

Store logs results into csv file

```
# Store logs into csv file
import pandas as pd
df_LR = pd.DataFrame.from_dict(cv_LR.cv_results_,orient='columns')
# Select columns to be stored
columns = ['params','mean_test_score','std_test_score','rank_test_score']
df_LR = df_LR[columns]
df_LR.to_csv("logs_LR.csv")
```

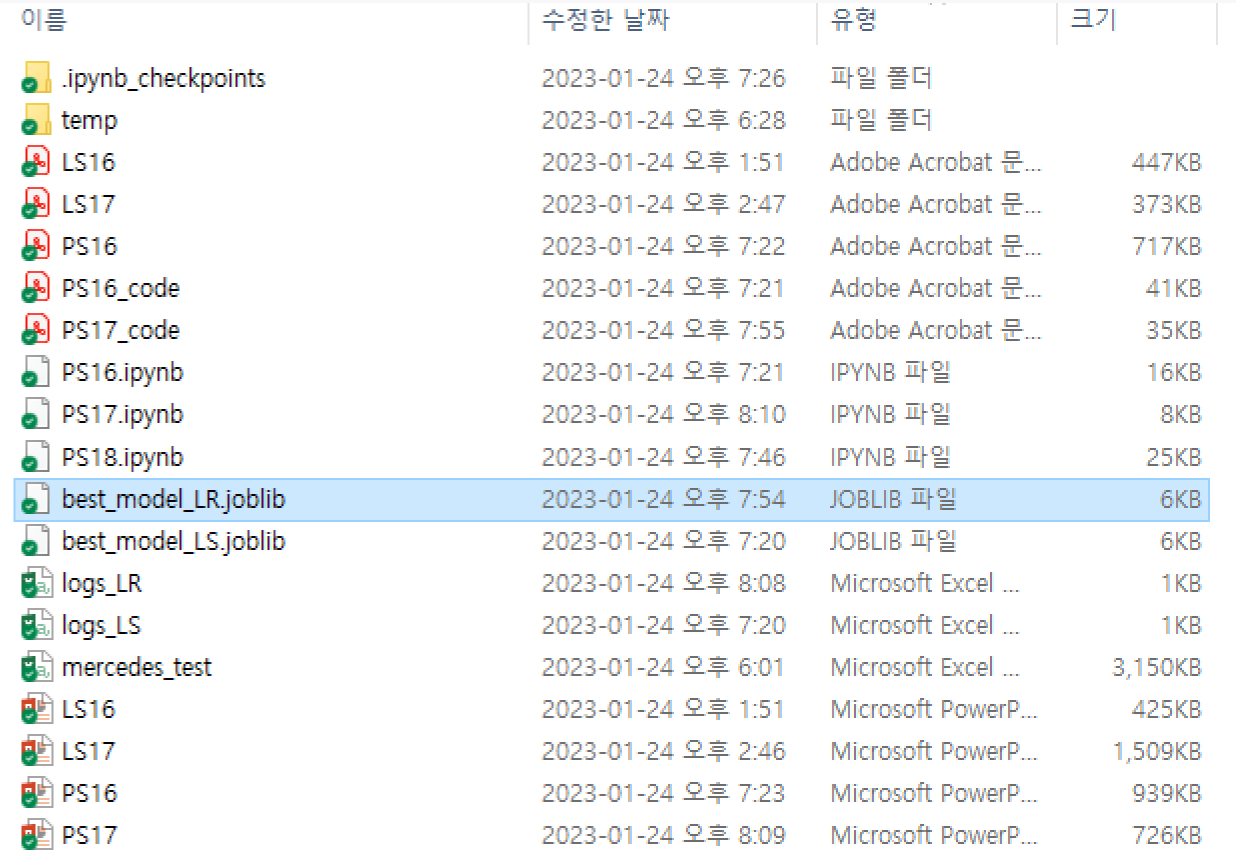
logs_LR.csv



	A	B	C	D	E
1		params	mean_test_score	std_test_score	rank_test_score
2	0	{'penalty': 'l2', 'C': 0.1}	0.883578771	0.008087593	1
3	1	{'penalty': 'l1', 'C': 0.1}	0.882786865	0.007955395	2
4	2	{'penalty': 'l1', 'C': 0.001}	0.500264201	0.000323579	5
5	3	{'penalty': 'l2', 'C': 0.001}	0.865362161	0.012273014	4
6	4	{'penalty': 'l1', 'C': 1}	0.880676047	0.006602031	3

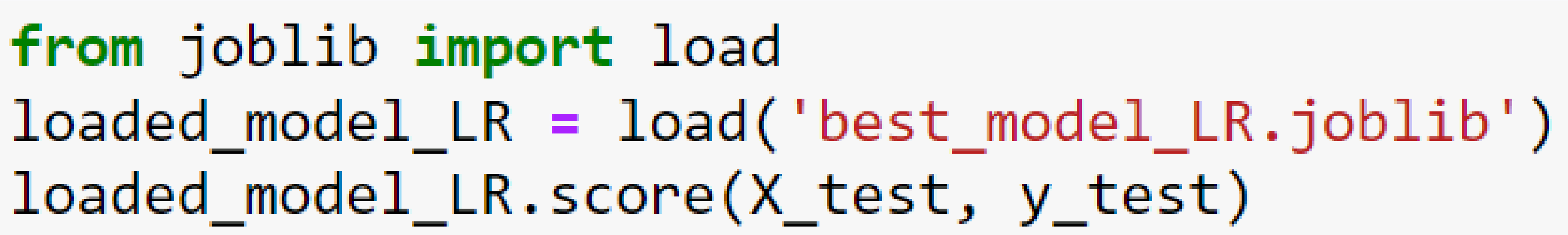
Save the best model

```python
best_model_LR=cv_LR.best_estimator_
from joblib import dump
dump(best_model_LR, 'best_model_LR.joblib')
```

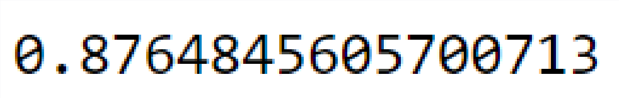


이름	수정한 날짜	유형	크기
.ipynb_checkpoints	2023-01-24 오후 7:26	파일 폴더	
temp	2023-01-24 오후 6:28	파일 폴더	
LS16	2023-01-24 오후 1:51	Adobe Acrobat 문...	447KB
LS17	2023-01-24 오후 2:47	Adobe Acrobat 문...	373KB
PS16	2023-01-24 오후 7:22	Adobe Acrobat 문...	717KB
PS16_code	2023-01-24 오후 7:21	Adobe Acrobat 문...	41KB
PS17_code	2023-01-24 오후 7:55	Adobe Acrobat 문...	35KB
PS16.ipynb	2023-01-24 오후 7:21	IPYNB 파일	16KB
PS17.ipynb	2023-01-24 오후 8:10	IPYNB 파일	8KB
PS18.ipynb	2023-01-24 오후 7:46	IPYNB 파일	25KB
best_model_LR.joblib	2023-01-24 오후 7:54	JOBLIB 파일	6KB
best_model_LS.joblib	2023-01-24 오후 7:20	JOBLIB 파일	6KB
logs_LR	2023-01-24 오후 8:08	Microsoft Excel ...	1KB
logs_LS	2023-01-24 오후 7:20	Microsoft Excel ...	1KB
mercedes_test	2023-01-24 오후 6:01	Microsoft Excel ...	3,150KB
LS16	2023-01-24 오후 1:51	Microsoft PowerP...	425KB
LS17	2023-01-24 오후 2:46	Microsoft PowerP...	1,509KB
PS16	2023-01-24 오후 7:23	Microsoft PowerP...	939KB
PS17	2023-01-24 오후 8:09	Microsoft PowerP...	726KB

Load “best_model_LR.joblib”



```
from joblib import load
loaded_model_LR = load('best_model_LR.joblib')
loaded_model_LR.score(X_test, y_test)
```



```
0.8764845605700713
```

Look ahead

Will implement DNN.