# 模型保存和加载的好处

## 不要每一次使用模型都得运行一下程序来生成模型,可以把我们满意的模型保存起来供有需要的时候使用

# 学习目标

|  |
| --- |
|  |

# 1.sklearn模型的保存和加载API

|  |
| --- |
|  |
|  |

# 2.线性回归的模型保存好加载案例

## 1>保存模型: 04\_boston-house-price-Ridge-save-model.py

|  |
| --- |
| """   波士顿房价预测,使用岭回归来实现   sklearn1.2以上版本没有load\_boston方法,需要使用fetch\_openml  """  from sklearn.datasets import fetch\_openml  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import Ridge  from sklearn.metrics import mean\_squared\_error  import joblib  # 1.获取数据集  data\_x, data\_y = fetch\_openml(name="boston", version=1, as\_frame=True,                                return\_X\_y=True, parser="pandas")  # 2 划分数据集  x\_train,x\_test,y\_train,y\_test = train\_test\_split(data\_x,data\_y,random\_state=22)  # 3.特征工程,标准化  transfer = StandardScaler()  x\_train = transfer.fit\_transform(x\_train)  x\_test  = transfer.transform(x\_test)  print(x\_train)  # 4.预估器流程  estimator = Ridge(alpha=0.2,max\_iter=9000)  # 5. 得出模型  estimator.fit(x\_train,y\_train)  # 5.2 保存模型  joblib.dump(estimator,"ridge.pkl")  print("岭回归-权重系数为：\n", estimator.coef\_)  print("岭回归-偏置为：\n", estimator.intercept\_)  # 模型评估  y\_predict = estimator.predict(x\_test)  print("岭回归-预测房价：", y\_predict)  error = mean\_squared\_error(y\_test,y\_predict) # 第一个参数是真实值,第二个参数是预测值  print("岭回归-均方误差：\n", error) |

### 运行后就会生成一个ridge.pkl模型

|  |
| --- |
|  |

## 2>加载模型: 05\_boston-house-price-Ridge-load-model.py

|  |
| --- |
| """   波士顿房价预测,使用岭回归来实现   sklearn1.2以上版本没有load\_boston方法,需要使用fetch\_openml  """  from sklearn.datasets import fetch\_openml  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import Ridge  from sklearn.metrics import mean\_squared\_error  import joblib  # 1.获取数据集  data\_x, data\_y = fetch\_openml(name="boston", version=1, as\_frame=True,                                return\_X\_y=True, parser="pandas")  # 2 划分数据集  x\_train,x\_test,y\_train,y\_test = train\_test\_split(data\_x,data\_y,random\_state=22)  # 3.特征工程,标准化  transfer = StandardScaler()  x\_train = transfer.fit\_transform(x\_train)  x\_test  = transfer.transform(x\_test)  print(x\_train)  # 4.预估器流程  # 把模型文件加载为模型  estimator = joblib.load("./ridge.pkl")  print("岭回归-权重系数为：\n", estimator.coef\_)  print("岭回归-偏置为：\n", estimator.intercept\_)  # 模型评估  y\_predict = estimator.predict(x\_test)  print("岭回归-预测房价：", y\_predict)  error = mean\_squared\_error(y\_test,y\_predict) # 第一个参数是真实值,第二个参数是预测值  print("岭回归-均方误差：\n", error) |

### 运行程序,效果如下

|  |
| --- |
|  |