Feature Engineering

数据嗨客

Feature Engineering

特征工程是将原始数据转换成特征的过程，这些特征能够很好的描述原始数据，或者使得机器学习模型性能达到最优.

符号标记	涉及包
X_train | 训练数据.	scikit-learn (0.19.1)
X_test | 测试数据.	scipy (0.19.0)
y_train | 训练集标签.	minepy (1.2.2)
y_test | 测试集标签.	

主要内容

数据预处理 | 将原始数据转换成可作为模型输入的数值型特征，包括数据标准化、离散化、特征编码等过程，也有的人将特征选择和降维归并在数据预处理之中. 请时刻谨记 Garbage In, Garbage Out.

特征选择 | 从原始特征全集中选择与目标特征相关的特征子集.

降维（特征提取） | 减少特征数量的过程，包括特征选择和特征提取，这里特指通过降维算法将多个原始特征进行组合得到新特征的特征提取过程，如主成分分析、线性判别分析等.

特征学习 | 利用机器学习算法或模型自动抽取特征的过程，如自编码器，并且大部分降维算法属于无监督特征学习算法.

数据预处理

```
from sklearn import preprocessing
```

方式1. 使用转换函数

```
X_scaled = preprocessing.scale(X_train)
```

方式2. 使用转换器接口（Transformer API）

```
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
```

```
scaler.fit_transform(X_train)
```

使用拟合-转换（fit+transform）一步到位的方法.

```
X_test_scaled = scaler.transform(X_test)
```

对测试数据做相同预处理.

标准化

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit_transform(X_train)
```

Z-score标准化	StandardScaler
最小最大标准化	MinMaxScaler
稀疏数据标准化	MaxAbsScaler
	RobustScaler
带离群值的标准化	QuantileTransformer
	RobustScaler

归一化

```
from sklearn.preprocessing import Normalizer
scaler = Normalizer()
scaler.fit_transform(X_train)
```

二值化

```
from sklearn.preprocessing import Binarizer
binarizer = Binarizer(threshold=0.0)
binarizer.fit_transform(X_train)
```

One-Hot编码

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse=False)
encoder.fit_transform(X_train)
```

特征取值必须为float类型或者int类型，不支持string类型.

目标特征编码

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
encoder.fit_transform(y_train)
```

二值化（产生多个二元特征）	LabelBinarizer
标签编码（不产生多个特征）	LabelEncoder
多标签二值化	MultiLabelBinarizer

缺失值填补

```
from sklearn.preprocessing import Imputer
imp = Imputer(strategy='mean', axis=0)
imp.fit_transform(X_train)
```

数据白化

```
from sklearn.decomposition import PCA
pca = PCA(2, whiten=True)
pca.fit_transform(X_train)
```

多项式特征生成与自定义转换

多项式特征生成

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
poly.fit_transform(X_train)
```

基于特征集合$\{X_1, X_2\}$，将产生新的特征集合$\{1, X_1, X_2, X_1^2, X_1X_2, X_2^2\}$.

自定义转换

```
from sklearn.preprocessing import FunctionTransformer
transformer = FunctionTransformer(np.log1p)
transformer.transform(X_train)
```

将所有数据进行对数转换.

特征选择

```
from sklearn import feature_selection as fs
```

过滤式（Filter）

按照特征与目标特征的相关性对各个特征进行评分，设定阈值或者待选择阈值的个数选择特征.

`fs.VarianceThreshold(threshold)` | 方差阈值过滤，去除特征全集中方差较小的特征.

`fs.SelectKBest(score_func, k)` | 保留得分排名前k的特征（top k方式）.

`fs.SelectPercentile(score_func, percentile)` | 保留得分排名前k%个特征（top k%方式），即最终保留特征的比例为k%.

```
from sklearn.feature_feature import SelectKBest, f_regression
filter = SelectKBest(f_regression, k=10)
filter.fit_transform(X_train, y_train)
```

使用F检验值对各个特征进行排名，并最终保留排名前10的特征作为最终的特征子集.

预测任务	评分指标	score_func
分类问题	卡方检验	chi2
	ANOVA F值	f_classif
	互信息MI	mutual_info_classif
回归问题	F检验值	f_regression
	互信息MI	mutual_info_regression

以下函数可以计算特征与目标特征的相关性，但是不能够作为score_func，需要进行转换.

`scipy.stats.pearsonr()` | 皮尔森相关系数.

`minepy.MINE().mic()` | 最大信息系数MIC.

封装式（Wrapper）

根据模型的评价指标，每次选择若干特征，或者排除若干特征，包括序列向前搜索SFS、序列向后搜索SBS等搜索方法，特征选择与最终的模型构建是两个独立的过程.

`fs.RFE(estimator, n_features_to_select)` | 递归特征消除法，得到指定特征个数的特征子集.

`fs.RFECV(estimator, scoring="r2")` | 结合交叉验证的递归特征消除法，得到特征子集的最优特征个数.

嵌入式（Embedded）

使用某些机器学习的算法（带正则化的线性模型、决策树以及基于决策树的集成模型）和模型自动进行特征选择，特征选择与模型评价融合在同一过程中.

`fs.SelectFromModel(estimator)` | 从模型中自动选择特征，任何具有coef_或者feature_importances_的基模型都可以作为estimator参数传入.

基模型	estimator
逻辑回归	linear_model.LogisticRegression
	linear_model.LogisticRegressionCV
LASSO	linear_model.Lasso
	linear_model.LassoCV
支持向量机	svm.LinearSVC
	svm.LinearSVR
随机森林	ensemble.RandomForestRegressor
	ensemble.RandomForestClassifier

示例

```
tree = RandomForestRegressor(n_estimators=100)
embeded = SelectFromModel(tree)
embeded.fit_transform(X_train, y_train)
```

降维（特征提取）

```
from sklearn import decomposition, manifold
```

主成分分析 | 通过线性投影，将高维的数据映射到低维的空间中，目标是最大化重构后方差.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit_transform(X_train)
```

线性判别分析 | 利用数据的类别信息，将高维的样本线性投影到低维空间中，使得数据样本在低维空间中的类别区分度最大，即使得相同类样本尽可能近，不同类样本尽可能远.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=2)
lda.fit_transform(X_train, y_train)
```

多维尺度变换 | 找到数据的低维表示，使得降维前后样本之间的相似度信息（如距离信息）尽量得以保留.

```
from sklearn.manifold import MDS
mds = MDS(n_components)
mds.fit_transform(X_train)
```

其他降维方法	对应的sklearn的类
核主成分分析	decomposition.KernelPCA
局部线性映射	manifold.LocallyLinearEmbedding
等度量映射	manifold.Isomap
t-SNE	manifold.TSNE
拉普拉斯映射	manifold.SpectralEmbedding

流水线处理（Pipeline）

```
from sklearn.pipeline import Pipeline
model = Pipeline([
    ('step1', StandardScaler()),
    ('step2', SelectFromModel(Lasso(alpha=5))),
    ('step3', RandomForestRegressor(100))
])
model.fit(X_train, y_train)
```

将数据预处理、特征选择、模型构建过程通过一个流水线串联起来.

Yan Xiaodong