一、机器学习流程、预处理、特征工程

1、三个主要页面（官网：http://scikit-learn.org/stable/index.html）：

（1）scikit-learn tutorials:

（2）User guide:

（3）API reference:

2、ml

（1）无监督方法：聚类、降维（PCA）

（2）有监督基本概念：

- Features 特征(属性、特征维、维度)
- Samples 样本
- Targets 目标值（y 值）

（3）预处理（Preprocessing）重点：

- 数据异常值处理（基本：箱线图的上界限和下界限筛选异常值；升级：根据数据分布做处理）等数据清洗
- 特征抽取（目标：对计算机读不懂的数据进行特征工程处理）

（4）模型：LR，GBDT，FFM，用 AOC 评价模型好坏

- 拓展：<1>把年龄小于 8 岁的打上“小孩”标签，年龄大于 60 岁打上“老人”标签，然后再做特征处理，将有很大的帮助=======
  <2>电商用户购买价格一般没有什么参考意义，但根据对场景的认识，做一定的统计处理，如根据买牛仔裤的平均价格预测下次买牛仔裤的价格
- 面试需准备的原理：朴素贝叶斯（文本类问题）——词袋模型（一句话中词的顺序不会影响大家对词的理解），抽取 TF-IDF，深度学习 LSTM；逻辑回归（万能简单的算法）；决策树（数学模型）随机森林/GBDT/XGBoost；SVM 统治了机器学习届 18 年（背后有严谨的数学支撑，如果不能手推 SVM，千万不要写精通 SVM，冯老师第八期课有手推 SVM）；k-means 聚类

（5）sklearn 常用方法（记）

- train_test_split：对数据做切分，test_size 一般使用 3/7 或 2/8 做数据切分，random_state 表示随机种子做采样或抽取
- 5 个函数（很重要!!!）：fit 做拟合，transform 对数据做降维等，fit_transform 既拟合也降维，predict 做预测，predict_proba 预测同时计算概率
- Perceptron
- 逻辑回归：$F(z)=\frac{1}{1+e^{-z}}$
- 支持向量机：rbf-kernel
- 决策树：通过机器学习找到判定的分支
- K 近邻（用得少）

（6）过拟合和正则化

- Overfitting：多灌数据、或加约束（例如 logistic 加 L1 或 L2 正则化项） ➔ 目的在于调整超参数
- Underfitting：换模型

二、Kaggle 机器学习比赛中的特征工程处理方法

1、如何分析数据：

- 先查看数据分布情况 info()、describe()
- 依次分析各个字段对结果的影响，然后拉出最突出的数据属性分析（简单，少用）

2、数据预处理（占比 70%的工作）

- 遇缺失值常见的处理方式：
  - 如果缺值的样本占总数比例极高，可能直接舍弃，避免带入 noise 影响最后的结果
  - 如果缺值的样本适中，而该属性为非连续值特征属性(比如说类目属性)，那就把 NaN 作为一个新类别，加到类别特征中
  - 如果缺值的样本适中，而该属性为连续值特征属性，有时候我们会考虑给定一个 step(比如这里的 age，我们可以考虑每隔 2/3 岁为一个步长)，然后把它离散化，之后把 NaN 作为一个 type 加到属性类目中。
  - 有些情况下，缺失的值个数并不是特别多，那我们也可以试着根据已有的值，拟合一下数据，补充上
- 对类目型的特征因子化/one-hot 编码，得到数值型特征的数据：可以使用 pandas 的"get_dummies"（哑变量变换）方法实现
- 标准化：将一些变化幅度较大的属性值特征化到[-1,1]之间：可以使用 scikit-learn 的 preprocessing 模块做一个 scaling（StandardScaler 模块）
- 归一化：scikit-learn.preprocessing 的 Normalization 模块
- 二值化：scikit-learn.preprocessing 的 Binarization 模块

3、用训练数据建模

- 使用 sklearn.model_selection 的 train_test_split 模块分割训练集和测试集，注意 X/Y 数据均是 numpy 格式的数据（pandas 的 dataframe 格式数据可以使用 df.as_matrix()方法转换）
- 常用模型：线性回归（sklearn.linear_model.LinearRegression）、logistic 回归（sklearn.linear_model.LogisticRegression）、SVM（sklearn.svm.SVC）、朴素贝叶斯（sklearn.naive_bayes.GaussianNB）、随机森林（sklearn.ensemble.RandomForestRegressor）、KNN（sklearn.neighbors.KNeighborsClassifier）

4、用测试集做 Prediction

- 注意：测试数据需要与训练数据保持一致的数据预处理再做 predict
- predict_proba()方法将输出有概率的预测

5、评估、优化模型

- 过拟合 V.S.欠拟合：
  - 过拟合(overfitting/high variace)：train 阶段表现好，test 阶段很不好
  - 欠拟合(underfitting/high bias)：train 阶段 test 阶段都不好
  - 解决：如果出现过拟合，可以做 feature selection 或者增加训练数据；如果出现欠拟合，通常需要更多的 feature 和更复杂的模型来提高准确度
- 使用 currentModel.coef_查看当前模型系数==========要对数据敏感了==========，通过观察权重系数分析各属性对结果的影响，例如：
  - 权重大将极大提高事件发生概率，反之，负相关将拉低事件发生概率
  - 小幅度正相关可能由于对属性细化程度不够=====试试将连续值离散化
- 做交叉验证优化模型，并绘制 learning_curve 曲线评判模型好坏

- 拿出原始训练集=====使用 sklearn.cross_validation.train_test_split 分割成 train 和 test 部分分别进行建模和预测，再观察原始数据中=====出现错误分类的数据=====，=======可以开始优化了=======
- 例如：组合两个属性成类目属性（采用 one-hot 编码）作为新的特征维度优化模型，去掉很奇怪但又没用的特征试试，blabla……
- 调参方法：

模型	使用示例
cross_validation	➢ Sklearn.cross_validation ➢ clf=LogisticRegression(C=1.0,penalty='l1',tol=1e-6) ➢ cross_validation.cross_val_score(clf, X, y, cv=5) ➔ 查看打分情况 ➢ split_train,split_cv=cross_validation.train_test_split(df, test_size=0.3, random_state=0) ➔ 分割训练集数据 ➢ clf.fit(split_train.as_matrix()[:,1:],split_train.as_matrix()[:,0]) ➢ predictions = clf.predict(split_cv.as_matrix()[:,1:]) ➢ split_cv[predictions != split_cv.as_matrix()[:,0]] ➔ 查看错误分类的数据 ➢ 特征选择、特征处理等
GridSearchCV	➢ sklearn.grid_search.GridSearchCV ➢ params={"n_neighbors":np.arange(1,3), "metric":{"euclidean","cityblock"}} ➢ grid = GridSearchCV(estimator=knn, param_grid=params) ➢ grid.fit(X_train,y_train) ➢ grid.best_score_ ➔ 得到交叉验证的平均结果 ➢ grid.best_estimator_.n_neighbors ➔ 寻找最佳参数
RandomizeSearchCV	➢ sklearn.grid_search. RandomizeSearchCV ➢ params={"n_neighbors":np.arange(1,5), "weights":{"uniform","distance"}} ➢ rsearch= RandomizeSearchCV(estimator=knn, param_distributions=param,cv=4,n_iter=8,random_state=5) ➢ rsearch.fit(X_train,y_train) ➢ rsearch.best_score_

➢ 模型融合

- 使用不同的分类器模型（logistic regression，SVM，KNN，random forest，神经网络），选择结果最优的模型
- 一个模型下的融合：使用 sklearn.ensemble.BaggingRegressor 模块，用训练集内不同子集训练出不同的模型，选择===最好/平均===，如

```
clf = linear_model.LogisticRegression(C=1.0, penalty='l1', tol=1e-6)
bagging_clf = BaggingRegressor(clf, n_estimators=10, max_samples=0.8, max_features=1.0,
                               bootstrap=True, bootstrap_features=False, n_jobs=-1)
bagging_clf.fit(X, y)
```

- 优点：缓解单一模型训练过程中产生的过拟合问题

三、特征工程处理方法、交叉验证、Pipeline 搭建机器学习模型

1、（很重要!!!）预处理（所有的库 ➔ sklearn.preprocessing: Preprocessing and Normalization）

- 独热向量编码（one-hot encoding）：OneHotEncoder 或者 pd.get_dummies()
- 连续值离散化：
  - bins = np.linspace(最小值，最大值，树桩数) ➔ 按树桩数划分数据区间
  - np.digitize(X，bins=bins) ➔ 使输入数据按照树桩分组
- 多项式特征（PolynomialFeatures）
  - 在原始数值列的基础上造些高次项（degree 属性），观察其与结果的线性关系（get_feature_names()方法查看新属性）
  - 构造组合特征

2、特征选择（sklearn.feature_selection）——绘制特征重要度的图，然后做特征筛选

（1）单变量分析（SelectPercentile 模块）：基于每一维对 y 的相关性进行特征选择

（2）基于模型的特征选择：SelectFromModel 模块可结合模型完成特征的选择

（3）逐步特征删除（RFE——Recursive feature selection 递归特征筛选）

（4）序列化特征选择（用得少）：mlxtend.feature_selection.SequentialFeatureSelector

3、模型评估与参数调优

（1）问题：直接使用 sklearn.model_selection.train_test_split 模块分割数据集存在随机性，对样本训练程度可能不够；

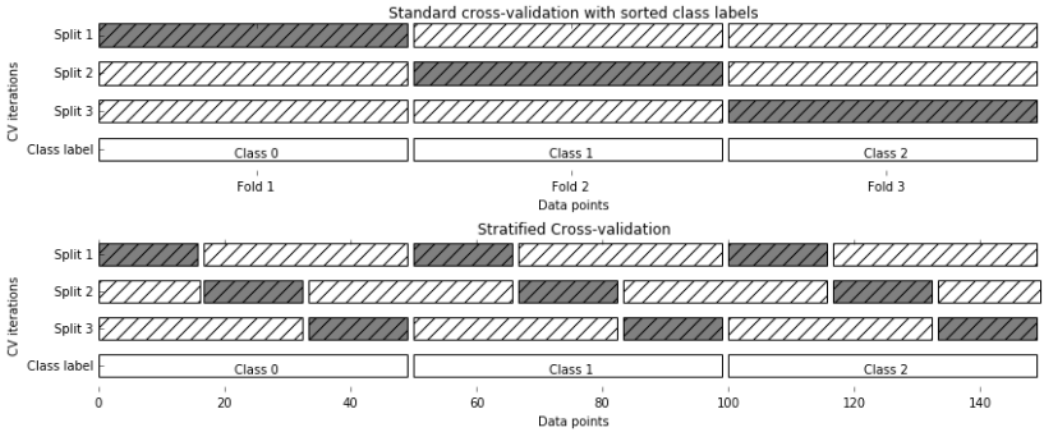
（2）交叉验证：

- sklearn.model_selection.cross_val_score 模块：
  - 默认 3 折（cv=3）交叉验证，返回测试集打分结果
  - 缺点：样本分布不均衡时，交叉验证效果不好，如下：

```
cross_val_score(logreg, iris.data, iris.target, cv=kfold)
```

```
array([ 1.        ,  0.93333333,  0.43333333,  0.96666667,  0.43333333])
```

- sklearn.model_selection.stratifiedkFold 模块：保证每一折中的样本比例均衡



- sklearn.model_selection.ShuffleSplit 模块：乱序分割交叉验证，赋给 cross_val_score 的 cv
- sklearn.model_selection.LeaveOneOut 模块：留一交叉验证，所有训练样本中留一个做交叉验证，赋给 cross_val_score 的 cv，基本不用

（4）网格搜索交叉验证（sklearn.model_selection.GridSearchCV 模块），为了找到最好的参数和最好的模型

- 给出参数字典 param_grid：
  - key 表示模型参数
  - value 表示所有候选值
  - 技巧：根据经验和对业务的理解设置参数，再通过二分法等继续调优参数
  - 调参指南表如下：

Model	Parameters to optimize	Good range of values
Linear Regression	• fit_intercept • normalize	• True / False • True / False
Ridge	• alpha • Fit_intercept • Normalize	• 0.01, 0.1, 1.0, 10, 100 • True/False • True/False
k-neighbors	• N_neighbors • p	• 2, 4, 8, 16 …. • 2, 3
SVM	• C • Gamma • class_weight	• 0.001, 0.01…..10...100...1000 • 'Auto', RS* • 'Balanced' , None
Logistic Regression	• Penalty • C	• L1 or l2 • 0.001, 0.01…..10...100
Naive Bayes (all variations)	NONE	NONE
Lasso	• Alpha • Normalize	• 0.1, 1.0, 10 • True/False
Random Forest	• N_estimators • Max_depth • Min_samples_split • Min_samples_leaf • Max features	• 120, 300, 500, 800, 1200 • 5, 8, 15, 25, 30, None • 1, 2, 5, 10, 15, 100 • 1, 2, 5, 10 • Log2, sqrt, None
Xgboost	• Eta • Gamma • Max_depth • Min_child_weight • Subsample • Colsample_bytree • Lambda • alpha	• 0.01,0.015, 0.025, 0.05, 0.1 • 0.05-0.1,0.3,0.5,0.7,0.9,1.0 • 3, 5, 7, 9, 12, 15, 17, 25 • 1, 3, 5, 7 • 0.6, 0.7, 0.8, 0.9, 1.0 • 0.6, 0.7, 0.8, 0.9, 1.0 • 0.01-0.1, 1.0 , RS* • 0, 0.1, 0.5, 1.0 RS*

- 初始化 GridSearchCV 完成参数的候选操作，如：GridSearchCV(SVC(), param_grid, cv=5)
- 用 fit 拟合数据集
- 得到交叉验证后的结果：
  - best_params_：最好的参数
  - best_score_：最高的评分
  - best_estimator_：最好的分类器，可以直接用
  - grid_scores_：查看所有交叉验证参数选择的结果

4、搭建流程（sklearn.pipeline）

（1）Pipeline 模块

- 用 list 初始化流水线上的所有环节，并给每个环节一个 tuple（名字，操作）
- 可以使用 GridSearchCV 找流水线上的最优参数，注意参数名称

（2）make_pipeline 模块

- 不需要指定流水线上每个环节的名字

- steps 属性：查看流程内的参数
- named_steps[name]：取出参数

四、机器学习建模实践

1、机器学习目的：从过去的大量数据中“总结”出来“泛化规律”，用于新数据预测。

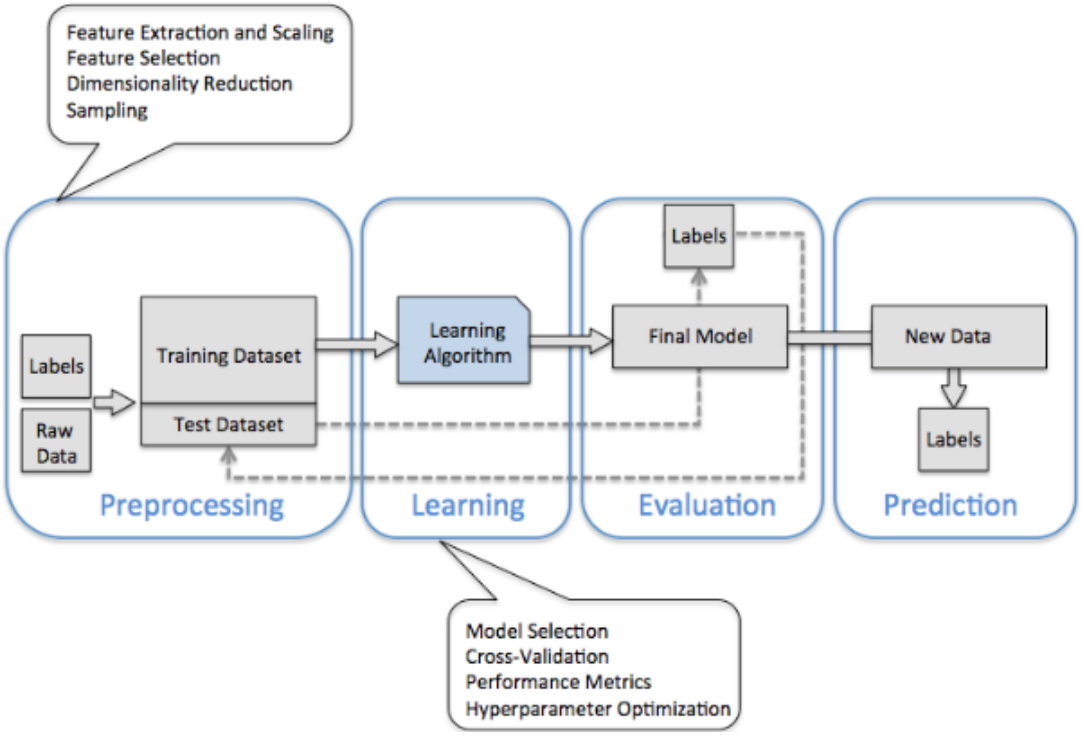
2、（面试）无监督学习：

➢ 聚类：k-means 怎么做？如何迭代？终止条件？如何确定团 k?

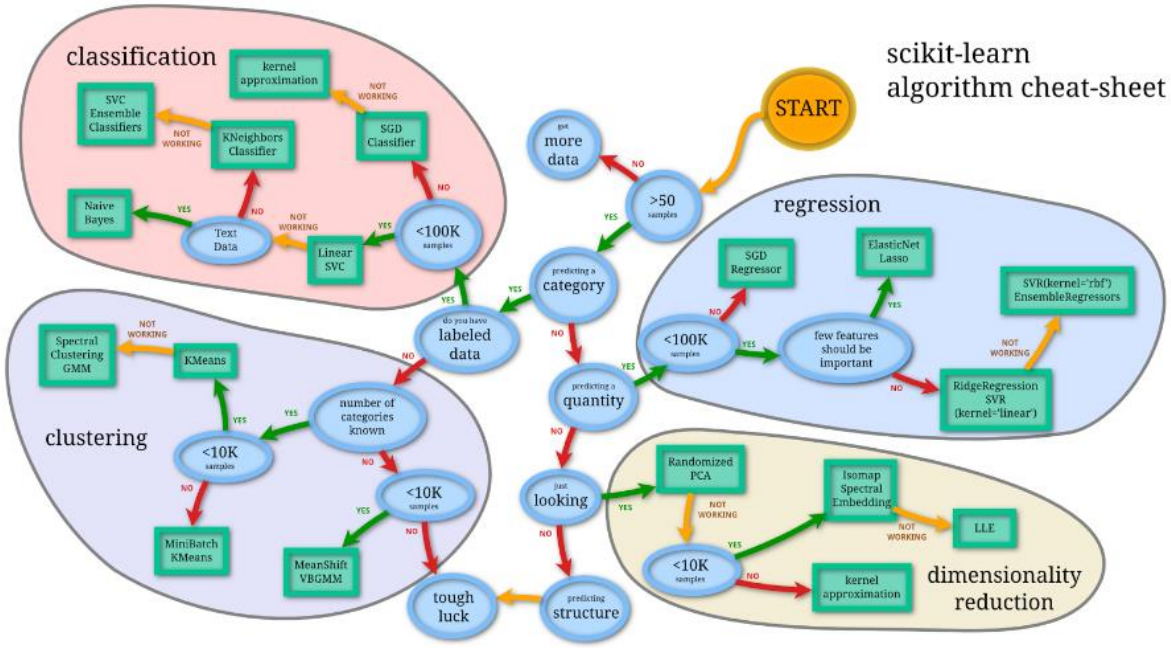
- 常用 k-means 方法：随机初始化 k 个中心点；对每个点找到最近的中心点并聚为一类，再对每一类更新中心点，重复迭代找到“最优的”中心点和 k 个类
- 优化：k-means++初始化法，随机选择第一个类中心点，再依次以“大概率”选择“距离已定中心点最近的点”作为下一个类中心点；
- 终止条件：由于目标函数单调递减且有界，所以一定收敛，能找到最优的中心点
- 确定团 k：使用交叉验证法、或肘方法（Elbow’s method）找到一个损失函数下降比较平的 k 值

➢ 降维：PCA

3、构建机器学习系统的一般流程



4、如何选择机器学习算法



5、自动特征选择

（1）sklearn.feature_selection.SelectPercentile 模块：按一定比例选择特征（存在随机性）

```
select = SelectPercentile(percentile=50)
select.fit(X_train, y_train)
# transform training set:
X_train_selected = select.transform(X_train)

print(X_train.shape)
print(X_train_selected.shape)
```

```
(284L, 80L)
(284L, 40L)
```

（2）sklearn.feature_selection.SelectFromModel 模块：基于模型的特征选择

```
select = SelectFromModel(RandomForestClassifier(n_estimators=100, random_state=42), threshold="median")
```

```
select.fit(X_train, y_train)
X_train_l1 = select.transform(X_train)
print(X_train.shape)
print(X_train_l1.shape)
```

```
(284L, 80L)
(284L, 40L)
```

（3）sklearn.feature_selection.RFE 模块：递归的删除不相关的特征

```
select = RFE(RandomForestClassifier(n_estimators=100, random_state=42), n_features_to_select=40)
#select = RFE(LogisticRegression(penalty="l1"), n_features_to_select=40)

select.fit(X_train, y_train)
```

五、使用 scikit-learn 刷天池 AI 电力能耗预测大赛

1、知识回顾

（1）常用模型

- 工业界：LR、DT、RF
- 比赛：xgboost、GBDT、NN

（2）LR：线性回归模型的 sigmoid 函数，输出每一特征维度的概率值，对分类/回归结果的相关性一目了然，模型的可解释性强，训练快，添加特征简单，常用于征信数据中。
（3）DT：空间体现即用很多超平面切割样本点，生成算法有 ID3（计算原树在新特征分割下的信息增益）、C4.5（信息增益率）、CART（基尼指数）。泛化能力差，极容易过拟合。
（4）SVM：使用 linear 核函数即弱化成线性模型，使用 rbf 核函数具有很高的模型复杂度，能解决非线性切分问题，但在数据量大的情况下计算非常复杂。
2、AI 电力能耗预测大赛实现步骤
（1）合并数据，包括训练集和测试集；
（2）统一做特征工程：如根据给的数据造些特征——day_of_week、day_of_month、day_of_year、month_of_year、holiday、week_of_month、period_of_month、festival 等
（2）拆分成训练集和测试集；
（3）选择合适的模型建模：LR、RF、GBDT 等；
（4）模型融合/网格搜索交叉验证调参；
（5）模型评估；
（6）实例主函数如下：

```
if __name__ == '__main__':
    print "Loding data..."
    total_df = load_data()
    print "Feature engineering..."
    total_df = feature_eng(total_df)
    print "Get features..."
    columns = get_columns(total_df)
    print "Split data..."
    train_X, test_X, train_y = split_data(total_df)
    print 'Create model...'
    # model = create_dt_model(train_X, train_y)
    # model = create_rf_model(train_X, train_y)
    model = create_xgb_model(train_X, train_y)
    print 'Predict model result...'
    result_df = predict_model(model, test_X)
    print 'Get feature importance...'
    feature_imp = get_feature_imp(train_X, model)
    plot_feature_imp(train_X, train_X.columns.values, model)
```