Data Analysis with Python

Cheat Sheet: Model Evaluation and Refinement

Process	Description	Code Example
| Splitting data for training and testing | The process involves first separating the target attribute from the rest of the data. Treat the target attribute as the output and the rest of the data as input. Now split the input and output datasets into training and testing subsets. | ```python
from sklearn.model_selection import train_test_split
y_data = df['target_attribute']
x_data=df.drop('target_attribute',axis=1)
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.10, random_state=1)
``` |
| Cross validation score | Without sufficient data, you go for cross validation, which involves creating different subsets of training and testing data multiple times and evaluating performance across all of them using the $R^2$ value. | ```python
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
lre=LinearRegression()
Rcross = cross_val_score(lre,x_data[['attribute_1']],y_data,cv=n)
# n indicates number of times, or folds, for which the cross validation is to be done
Mean = Rcross.mean()
Std_dev = Rcross.std()
``` |
| Cross validation prediction | Use a cross validated model to create prediction of the output. | ```python
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LinearRegression
lre=LinearRegression()
yhat = cross_val_predict(lre,x_data[['attribute_1']], y_data,cv=4)
``` |
| Ridge Regression and Prediction | To create a better fitting polynomial regression model, like , one that avoids overfitting to the training data, we use the Ridge regression model with a parameter alpha that is used to modify the effect of higher-order parameters on the model prediction. | ```python
from sklearn.linear_model import Ridge
pr=PolynomialFeatures(degree=2) x_train_pr=pr.fit_transform(x_train[['attribute_1', 'attribute_2', ...]])
x_test_pr=pr.fit_transform(x_test[['attribute_1', 'attribute_2',...]])
RigeModel=Ridge(alpha=1)
RigeModel.fit(x_train_pr, y_train)
yhat = RigeModel.predict(x_test_pr)
``` |
| Grid Search | Use Grid Search to find the correct alpha value for which the Ridge regression model gives the best performance. It further uses cross-validation to create a more refined model. | ```python
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
parameters= [{'alpha': [0.001,0.1,1, 10, 100, 1000, 10000, ...]}]
RR=Ridge()
Grid1 = GridSearchCV(RR, parameters1,cv=4) Grid1.fit(x_data[['attribute_1', 'attribute_2', ...]], y_data)
BestRR=Grid1.best_estimator_
BestRR.score(x_test[['attribute_1', 'attribute_2', ...]], y_test)
``` |