 Message

*From: Guido Rossum*

Building the Models

Hi,

Now that you have successfully imported, prepared and explored the data you are ready to start exploring some possible tools for your analysis. In this task you’ll build your models just as you have done previously. As you progress remember the following:

1. Let the data tell the story – don't make any assumptions.
2. It is often best to build three or more models and compare the results.
3. Make sure you have chosen the correct tools for the type of data you have.
4. Be prepared to investigate possible alternative solutions in case your methods do not work as expected.

I suggest you start this task with a quick review of Sci-Kit Learn to ensure you are familiar with the benefits of using it and how to use it effectively for this project.

GR

**Guido Rossum**

Senior Data Scientist

Credit One

www.creditonellc.com

Project Roadmap

Now that you have properly prepared and thoroughly explored the data it's time to begin the modeling process. Throughout this task will examine feature selection and model building. Is very important for you to understand that this task uses the CreditOne data in a regression type problem, which is different from what you did in course one. The steps will be very similar, but you will need to replicate and them in a different way and obviously on different features and variables. Let's get started with a quick review of Sci-Kit Learn.



1

Selecting and Dividing the Data

2

Building the Models and Choosing the Right Model

3

Making Predictions and Evaluating the Results

4

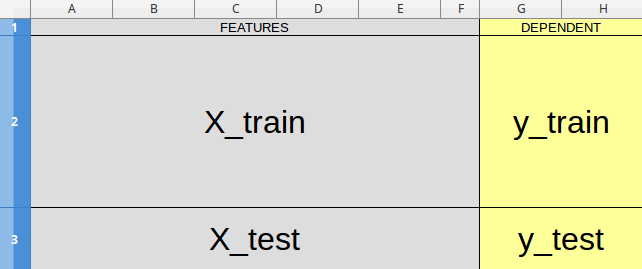
Complete the CreditOne Project and Writing Your Report

1. Selecting and Dividing the Data

Review of Sci-Kit Learn

Remember that everything you will be doing in this task is centered around a regression problem using the CreditOne data. This regression problem here is being introduced to walk you through learning 'how' to use Sci-Kit Learn, but you'll likely use a different dependent variable for your final analysis.

Data Structure

This format is the same as you've experienced so far especially in course one:  
  


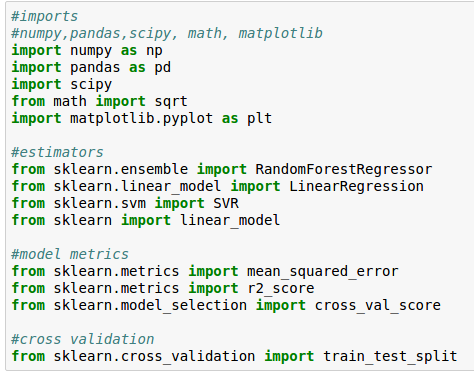
Here is a reminder of the description of each partition and how it resembles what you already know from your previous work (in Course one):

* Training Data is divided into two parts: one set for the features (X-values) and a related set for the outcomes (y-values).
  + Both of the should match your training split size (75%, for example).
* Testing data follows the same rules and contains two sets: one for the X-values or features and a second for the ground truth (y\_test), which you'll use to compare your predictions to.
  + Testing data is composed of what is left over from the training data (25%, for example).

Lets import the modules and the data. In a few more steps we'll select the features and the divide the data into training and testing sets:

**1.Start a new notebook and save it as CreditOne Regression**

**2. Import the modules you'll need for this task as shown below (we'll learn more about some of these later):**



**3. Import the data**

If you've created a new .csv file containing your pre-processed and cleaned data do not forget to use it instead of the original!

#data  
rawData = pd.read\_csv('default of credit card clients.csv', header=1)  
rawData.head()

**4. Examine the structure of your data and ensure everything was imported in the format you need for work.**

rawData.info()

Selecting Data

Let us select the features and the dependent variable:

**1. Select the features (you should now have determined a final set of features to be used in your analysis after studying the correlation and covariance of the data):**

#features  
X = rawData.iloc[you can choose what goes here]  
print('Summary of feature sample')  
X.head()

**2. Select the Dependent Variable (Remember - this is just for this example; your final model might need to use something different):**

#dependent variable  
y = rawData*[This is up to you!]*

Sci-Kit Learn Format (Review)

Now that we've selected the features an established the dependent variable, we can begin to use Sci-Kit Learn; the module contains many embedded functions and methods we can 'call' or use whenever we need to. One of the most often used functions in machine learning is obviously an algorithm. We will access algorithms though one of Sci-Kit Learn included objects called the Estimator Object.

Lets examine this concept a little more:

The Sci-Kit Learn Estimator Object can be accessed just like any other library by importing it into Python (your notebook) as follows:

from sklearn

Note that we're not simply using import as we've done previously. This is because we need to specify where the needed function "lives" before importing it. ***sklearn is a portion of the Estimator Object***, the remainder is specific to the algorithm needed, and this is where the ***import*** function is used as follows (using Linear Regression as an example):

from sklearn.linear\_model import LinearRegression

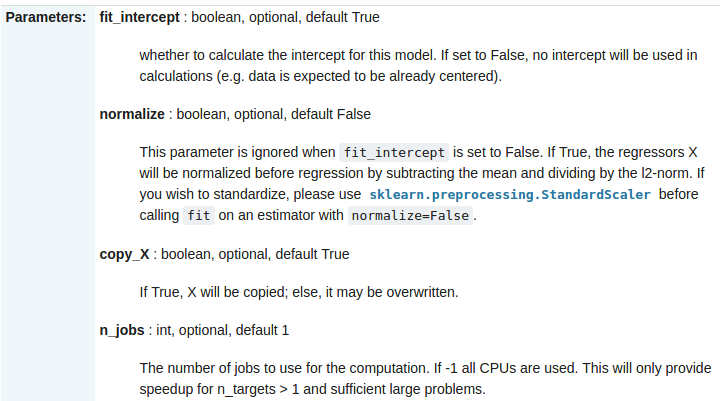
As you can see, the linear\_model Estimator Object was used to access the *LinearRegression* algorithm from the ***sklearn*** base. This is how almost all algorithms are used or 'called' from Sci-Kit Learn. You can read more about the Estimator Object in the resources.

Tuning Parameters and Model Definition

As you already know, almost all algorithms have associated tuning parameters. Previously, you accessed these parameters from with R packages or from with caret, but the parameters in Sci-Kit Learn models can be easily accessed when defining the model itself. Here is an example of defining the model and accessing its parameters:

model = LinearRegression(parameters go here)

In this example LinearRegression has the following parameters ([from the Sci-Kit Learn Reference](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression)):



Any of the above parameters can be accessed when defining the *LinearRegression* model; here is an example of tuning the number of 'n-jobs' that should be used for the computation:

model = LinearRegression(n\_jobs=10)

For most uses this is how parameters can be tuned and how you'll build and refine the model, which will be used later for fitting and making predictions just like you've done previously in course one. Consult the resources for more information on any of the parameters associated with algorithm.

Ok, now our data is ready; lets build some models!

Next, we'll dive into 'how' to choose the best model for our data from three different algorithms and apply the most optimal one to make predictions.

 2. Building the Models and Choosing the Right Model

Now that you have reviewed using Sci-Kit Learn, let's walk through the process of building a pipeline for creating a model and making predictions with Python and Sci-Kit Learn. During this step we'll build a three Regression models and choose the right one for our needs. After working through the next few steps you will be ready to build your own models for the Credit One task Guido has assigned to you using Python for Data Science.

**1. If you haven't already done so start by importing the modules we need for this step in a new notebook:**

#imports  
#numpy,pandas,scipy, math, matplotlib  
import numpy as np  
import pandas as pd  
import scipy  
from math import sqrt  
import matplotlib.pyplot as plt

#estimators  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.linear\_model import LinearRegression  
from sklearn.svm import SVR

#model metrics  
from sklearn.metrics import mean\_squared\_error  
from sklearn.metrics import r2\_score  
from sklearn.model\_selection import cross\_val\_score

#cross validation  
from sklearn.cross\_validation import train\_test\_split

**2. Again while not mandatory, but it might be a good idea to validate that the correct column is being used for the dependent variable:**

#dependent variable  
print(y)

**Cross Validation and Choosing the Right Model**

It is often necessary to build more than one model in order to find the most appropriate one for the job. Sci-Kit Learn has a function that will 'score' each model for appropriateness based on the algorithm that was used to build it; models with the highest scores should be used to make predictions. The metric we will use to check for proper model usage is called the *cross\_val\_score.*

To use this function, we need to verify the library has been imported as needed. Examine the cell of your notebook where your imports live and verify the following has been added or is already present:

from sklearn.model\_selection import cross\_val\_score

Then simply pass each trained model and the training sets to the function to obtain the training score. Here is an example of what the code looks like:

print(cross\_val\_score(model, X, y, cv=3))

and it returns three values in a Python Dictionary as follows (just an example):

These values correspond to each score in an array of scores of the estimator for each run of the cross validation. The higher the average, the better the estimator will perform.

**Modeling Format:**

**1. Lets go ahead and prepare each regression algorithm for use just as we did in course one**

algosClass = []

algosClass.append(('Random Forest Regressor',RandomForestRegressor()))

algosClass.append(('Linear Regression',LinearRegression()))

algosClass.append(('Support Vector Regression',SVR()))

**2. To build and assess both models recursively, we create an empty list to store the results and another to hold the name of each algorithm so we can easily print out the results and keep them separated as follows:**

#regression  
results = []  
names = []  
    for name, model in algosClass:  
    result = cross\_val\_score(model, X,y, cv=3, scoring='r2')  
    names.append(name)  
    results.append(result)

Notice how we're passing the *cross\_val\_score* within the for loop? This will test all three models using three different 'folds' of the data and R Squared (r2) (more on this in just a bit) as the assessment criteria). Here is the output:

for i in range(len(names)):  
    print(names[i],results[i].mean())

* Random Forest Regressor 0.4233078518164956
* Linear Regression 0.3581989426610932
* Support Vector Regression -0.05617731750522129

Note: Your numbers might  greatly from these results since you have already selected the features you want to use.

Based on what you've already learned about cross validation score, choose the best model that you will use in the next step to train the model make predictions.

**3. Use the model variables you established in step 2, pass the training data to it in the following format (you'll need to use train\_test\_split prior):**

algo = choose your algorithm()

model = algo.fit(X\_train,y\_train)

Now you have tested multiple options and you have chosen one to use and trained it; lets move on to making predictions.

3. Making Predictions and Evaluating the Results

Making Predictions

Now that you have trained a model, let's use it to make predictions. This process is almost identical to what you have done in course one (here is an example):

predictions = modelRF.predict(X\_test)

**Pre-work**

Evaluting Regression problems is a bit different than assessing classification problems; we're doing the same thing (measuring), but we use different metrics to do so. Before you dive into this next section, it is a good diea to have a quick read of the following article:

[[](https://s3.amazonaws.com/gbstool/courses/1095/docs/How%20to%20evaluate%20regression%20models.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20201223T162733Z&X-Amz-SignedHeaders=host&X-Amz-Expires=36900&X-Amz-Credential=AKIAJBIZLMJQ2O6DKIAA%2F20201223%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=698939093fbc4c8231b3ea6d2399b0988cbc2d1671ce809bd4c91f71f171bd8a) How to evaluate regression models](https://s3.amazonaws.com/gbstool/courses/1095/docs/How%20to%20evaluate%20regression%20models.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20201223T162733Z&X-Amz-SignedHeaders=host&X-Amz-Expires=36900&X-Amz-Credential=AKIAJBIZLMJQ2O6DKIAA%2F20201223%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=698939093fbc4c8231b3ea6d2399b0988cbc2d1671ce809bd4c91f71f171bd8a) *Vimarsh Karbhari. (2018, December 18). How to evaluate regression models? Medium; Acing AI. https://medium.com/acing-ai/how-to-evaluate-regression-models-d183b4f5853d*

Evaluating the Results

**1. The sklearn.metrics Object is the main object that contains almost all of the metric functions you will need. Verify that the first two are in your imported list of libraries (you'll see some familiarity in their names):**

from sklearn.metrics import mean\_squared\_error  
from sklearn.metrics import r2\_score

Since this is a Regression problem you will use RMSE and R squared to measure the trained model.

‌So why didn't you import a metric called RMSE? RMSE is just the Square Root of MSE (mean squared error) so you will also need to ensure that we've also included the sqrt function from the math library during import:

from math import sqrt

**2. Now you can use the sqrt function and the mean\_squared\_error function to compose your own function for calculating RMSE:**

rmse = sqrt(mean\_squared\_error(y\_test, predictions))

**3. Next, establish a variable and use the included function, the ground truth, and the predictions to calculate R Squared as follows:**

predRsquared = r2\_score(y\_test,predictions)

**4. Here is how it all looks together:**

#Make Predictions  
predictions = model.predict(X\_test)  
predRsquared = r2\_score(y\_test,predictions)  
rmse = sqrt(mean\_squared\_error(y\_test, predictions))  
print('R Squared: %.3f' % predRsquared)  
print('RMSE: %.3f' % rmse)

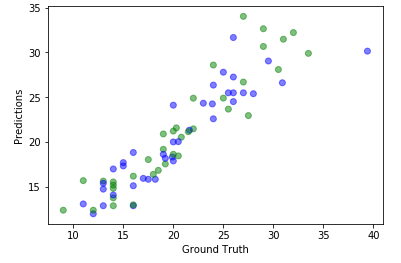
 Plotting the Results

The final step in the model building process (unless the model will be deployed) is often to plot a comparison between the known values in the test set and the predictions made by the model.

**5. To do this we can create a simple scatter plot using matplotlib as we've done previously.**

plt.scatter(y\_test, predictions, color=['blue','green'], alpha = 0.5)  
plt.xlabel('Ground Truth')  
plt.ylabel('Predictions')  
plt.show();

This produces the following plot (this is just an example):



Questions:

1. Did any of the algorithms provide reasonable or acceptable results? Why or why not?
2. Can this problem be addressed or approached in a different manner that might work better?
   1. Is there some way to still use credit limit as the dependent variable, but solve this as a different machine learning problem?
   2. If not, what can be done to satisfy the client's needs and business scope? (Hint: They listed a minimum requirement).

Mentor's Note: This course is designed to challenge your assumptions, build your confidence and make you think out of the box about possible creative solutions so do not be surprised if the 'normal' process doesn't work quite as well as you would expect it to. Remember: There is 'no free lunch' in Data Science!

4. Complete the CreditOne Project and Writing Your Report

Now you are ready to dive into the CreditOne data and apply what you've already learned; much of the work you have already done can be applied to this task so know you're not starting completely from the beginning.

One of the main objectives that separates Data Science from Data Analytics is creative problem-solving skills and many times Data Scientists will have to learn new skills just to solve a problem; you'll be doing the same thing here and stretching your existing knowledge by learning things that you have not been introduced to yet to solve the problem at hand.

In addition to applying your new Python and Sci-Kit Learn skills you will also need to demonstrate that you can use the Sci-Kit Learn resources to perform some additional Data Science related skills.

Here is the list of requirements that your data science process should include for your final report:

1. Cleaning and [Pre-processing](http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing)
2. EDA
3. [One-Hot Encoding](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html)(if needed)
4. Regression (Build three model and choose the best)
5. Model Evaluation

You can use the learning resources to assist you with your work.

**This task requires that you prepare one deliverable and one Juypter Notebook:**

1. Customer Default Identification Report that addresses:

Problem:

An increase in customer default rates is bad for Credit One since its business is approving customers for loans in the first place. This is likely to result in the loss of Credit One's business customers. You need to build a model that can better predict what credit limit a customer should be assigned.

Questions to Investigate:

1. How do you ensure that customers can/will pay their loans?
2. Can we approve customers with high certainty?

As you progress through the task, begin thinking about how to solve the company's problem.

Here are some lessons the company learned from addressing a similar problem last year:

1. We cannot control customer spending habits
2. We cannot always go from what we find in our analysis to the underlying "why"
3. We must focus on the problems we can solve:
   1. Which attributes in the data can we deem to be statistically significant to the problem at hand?
   2. What concrete information can we derive from the data we have?
   3. What proven methods can we use to uncover more information and why?

Guido is expecting a report in a few days:

1. Your report should be a one to three page Word document that includes rules you believe provide insights, any relevant visualizations, and the answers to the company's questions.
2. It should also include any observations that you've made and any recommendations you might have, supported by evidence uncovered in your analysis.

2. Your Data Science work should be submitted as a Juypyter Notebook and in your GitHub account.

This task requires that you prepare one deliverable and one Juypter Notebook:

Customer Default Identification Report that addresses:

Problem:

An increase in customer default rates is bad for Credit One since its business is approving customers for loans in the first place. This is likely to result in the loss of Credit One's business customers.

Questions to Investigate:

1. How do you ensure that customers can/will pay their loans?
2. Can we approve customers with high certainty?

As you progress through the task, begin thinking about how to solve the company's problem.

Here are some lessons the company learned from addressing a similar problem last year:

1. We cannot control customer spending habits
2. We cannot always go from what we find in our analysis to the underlying "why"
3. We must focus on the problems we can solve:
   1. Which attributes in the data can we deem to be statistically significant to the problem at hand?
   2. What concrete information can we derive from the data we have?
   3. What proven methods can we use to uncover more information and why?

Guido is expecting a report in a few days:

1. Your report should be a one-to-three-page Word document that includes rules you believe provide insights, any relevant visualizations, and the answers to the company's questions.
2. It should also include any observations that you've made and any recommendations you might have, supported by evidence uncovered in your analysis.
3. Your Data Science work should be submitted as a Juypyter Notebook and in your GitHub account.