Modelo de Regresion Seleccion de Caracteristica Usando LASSO

En este Libro se utiliza LASSO para seleccionar las caracteristicas. se correra LASSO con diferentes L1 que son los reguladores y de todos los L1 con los que se ejecute LASSO se escogera el que mas bajo RSS obtenga con la data de validacion.

Importacion de todas las librerias necesarias

In [66]:

```
#libreria para importar las funciones que facilitan la creacion de modelos usando regresion
import turicreate
from math import log, sqrt
#libreria para realizar calculos matriciales
import numpy
import matplotlib.pyplot as plt
%matplotlib inline
```

Carga de la sabana de datos con la información de las notas de los estudiantes

La información fue proporcionada por el cordinador del Colegio Almirante Padilla, Colegio ubicado en la localidad quinta de Usme.

In [6]:

```
#El tipo de dato de cada columna que se va a cargar.
column_type_hints=[str,str,str,str,str,str,str,float,float,float,float,float,float,float,float,floa
t,float,float,float,float,float,float,float,str,str,str,str]
rEstudiante=turicreate.SFrame.read_csv('DataEstudiantes.txt','&',column_type_hints=column_type_hint
s)
len(rEstudiante)
```

```
Finished parsing file
/home/santos/MachineRegression/Week4/Assigment1/DataEstudiantes.txt


Parsing completed. Parsed 214 lines in 0.03368 secs.
```

Out[6]:

```
214
```

Se cargaron 214 registros.

Selección de Estudiantes

Se toman aquellos estudiantes que tiene una nota de calculo mayor a 0, este proceso se realiza ya que los registros que no tienen una nota de calculo son de estudiantes que no continuaron en el colegio.

In [5]:

```
rEstudiante=rEstudiante[rEstudiante['CALCULO']>0]
len(rEstudiante)
```

Out[5]:

```
182
```

- Despues del Filtro quedaron 182 registros.

Carga Sabana de Datos para Evaluar Modelos.

Esta Data no interviene para la generacion de los coeficientes del modelo, es utilizada para la evaluacion de la prediccion del modelo.

In [12]:

```
column_type_hints=[str,str,str,str,str,str,str,float,float,float,float,float,float,float,float,floa
t,float,float,float,float,float,float,float,float,str,str,str,str]
evaluation=turicreate.SFrame.read_csv('DataEvaluacion2018.txt','&',column_type_hints=column_type_hi
nts)
evaluation=evaluation[evaluation['CALCULO']>0]
```

```
Finished parsing file
/home/santos/MachineRegression/Week4/Assigment1/DataEvaluacion2018.txt


Parsing completed. Parsed 84 lines in 0.034066 secs.
```

Se utilizaron 84 registros para evaluar el modelo

Seleccion Caracteristicas Inicial

Para comenzar con la seleccion de las caracteristicas se inicia el modelo considerando todas.

In [8]:

```
all_features=['ECONOMIA','FÍSICA','QUÍMICA','SOCIALES','EDUCACION ARTISTICA','RECREACIÓN Y
DEPORTES','RELIGION','ETICA','FILOSOFIA','INGLES','LENGUA
CASTELLANA','TRIGONOMETRIA','ADMINISTRACION','MANEJO  ORDENADORES','PREPARACIÓN TECNOLÓGICA']
```

Seleccionando L1 Penalizacion

Para encontrar el mejor valor de L1, se exploraron multiples valores usando el conjundo de datos de validacion. De el conjunto de datos cargado se realizo tres divisiones. datos de entrenamiento, datos de validacion y datos de prueba.

In [9]:

```
(training_and_validation, testing) = rEstudiante.random_split(.9,seed=1) # el 90% de datos para ent
renar y validar
(training, validation) = training_and_validation.random_split(0.5, seed=1) # 50% para entrenar y 50
% para validar.
```

La funcion retorna la ecuacion con los coeficientes.

In [175]:

```
def FormarEcuacion(modelo):
    all_features=["(intercept)",'ECONOMIA','FÍSICA','QUÍMICA','SOCIALES','EDUCACION
ARTISTICA','RECREACIÓN Y DEPORTES','RELIGION','ETICA','FILOSOFIA','INGLES','LENGUA CASTELLANA','TR
IGONOMETRIA','ADMINISTRACION','MANEJO  ORDENADORES','PREPARACIÓN TECNOLÓGICA']
    ecuacion=""
    contador=0
    for i in modelo.coefficients['value']:
        if contador==0:
                ecuacion+=str(i)
        else:
            ecuacion+=" + "+str(i)+" H"+str(contador)+"(x)"
        contador+=1
    return ecuacion
        #print (i+" "+str(Rl1_penalty[0][i]))
```

Las siguientes funciones obtienen los coeficientes del modelo y almacenan el modelo encontrado por cada L1 que se pruebe.

In [176]:

```
#Funcion para guardar el registro de los coeficientes y el RSS generado por los datos de entrenami
ento, validacion y evaluacion.
def InsertarRLambda(model,L,RSSTrain,RSSValidation,RSSTest,RSSEvaluation):
    lambdas=turicreate.SFrame({'λ':[L],'ECONOMIA':[float()],'FÍSICA':[float()],'QUÍMICA':
[float()],'SOCIALES':[float()],'EDUCACION ARTISTICA':[float()],
                               'RECREACIÓN Y DEPORTES':[float()],'RELIGION':[float()],'ETICA':[flo
()],'FILOSOFIA':[float()],'INGLES':[float()],'LENGUA CASTELLANA':[float()],
                               'TRIGONOMETRIA':[float()],'ADMINISTRACION':[float()],'MANEJO  ORDEN
ORES':[float()],'PREPARACIÓN TECNOLÓGICA':[float()],'(intercept)':[float()]
                              ,'RSSTrain':[float()],'RSSValidation':[float()],'RSSTest':
[float()],'RSSEvaluation':[float()],'ecuacion':[str()]})
    lambdas['RSSTrain']=RSSTrain
    lambdas['RSSValidation']=RSSTrain
    lambdas['RSSTest']=RSSTrain
    lambdas['RSSEvaluation']=RSSEvaluation
    coeficientes=model.coefficients
    for i in range(len(coeficientes)):
        lambdas[coeficientes[i]['name']]=coeficientes[i]['value']
    lambdas["ecuacion"]=FormarEcuacion(model)
    return lambdas

#Funcion que obtiene los coeficientes para un L1
def lasso_regression(rEstudiante, features, l1_penalty):
    model = turicreate.linear_regression.create(rEstudiante, target='CALCULO', features=features,
                                                validation_set=None,
                                                l2_penalty=0., l1_penalty=l1_penalty,verbose=False)

    return model
```

Implementacion de LASSO

Se realizan 2000 iteraciones con diferentes L1 para obtener el RSS usando la data de validacion.

In [181]:

```
lModel=[]
Rl1_penalty=turicreate.SFrame()
flag=True
for l1_penalty in np.logspace(1, 10, num=2000):#np.logspace(1, 7, num=13):
    model=lasso_regression(training, all_features, l1_penalty)
    lModel.append((l1_penalty,model))
    RSSTrain=((model.predict(training)-training['CALCULO'])**2).sum()
    RSSValidation=((model.predict(validation)-validation['CALCULO'])**2).sum()
    RSSTest=((model.predict(testing)-testing['CALCULO'])**2).sum()
    RSSEvaluation=((model.predict(evaluation)-evaluation['CALCULO'])**2).sum()
    if flag==True:
        Rl1_penalty=InsertarRLambda(model,l1_penalty,RSSTrain,RSSValidation,RSSTest,RSSEvaluation)
        flag=False
    else:
        Rl1_penalty=Rl1_penalty.append(InsertarRLambda(model,l1_penalty,RSSTrain,RSSValidation,RSST
est,RSSEvaluation))
```

Graficas

Las siguiente funciones sirven para graficar el modelo

In [182]:

```
def plot_coeficientes(data,X,Y,c,scx,scy,i,ylabel):
    if i==0:
        plt.figure(figsize=(10,10))
        plt.xlabel(X)
        plt.ylabel(ylabel)
    plt.plot(data[X], data[Y], c, label=Y)
    plt.legend(loc='upper right')
    plt.yscale(scx)
    plt.xscale(scy)

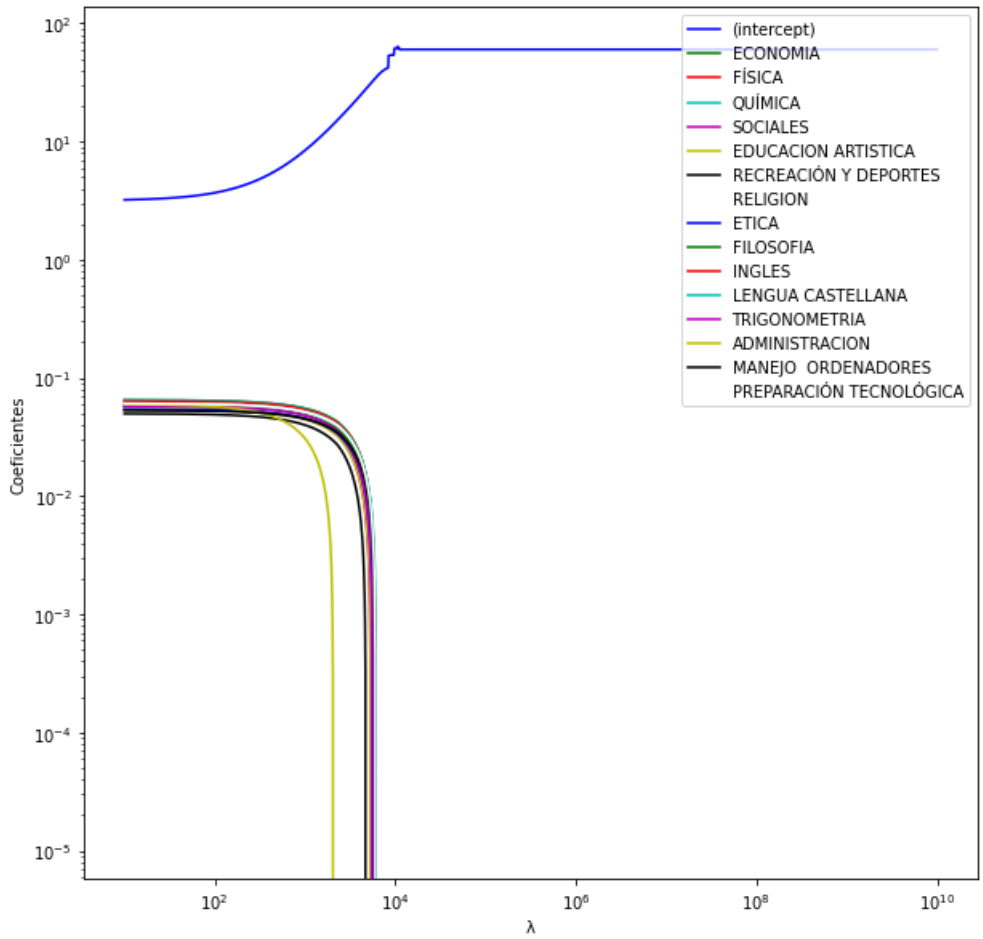
all_features1=["(intercept)"]+all_features
```

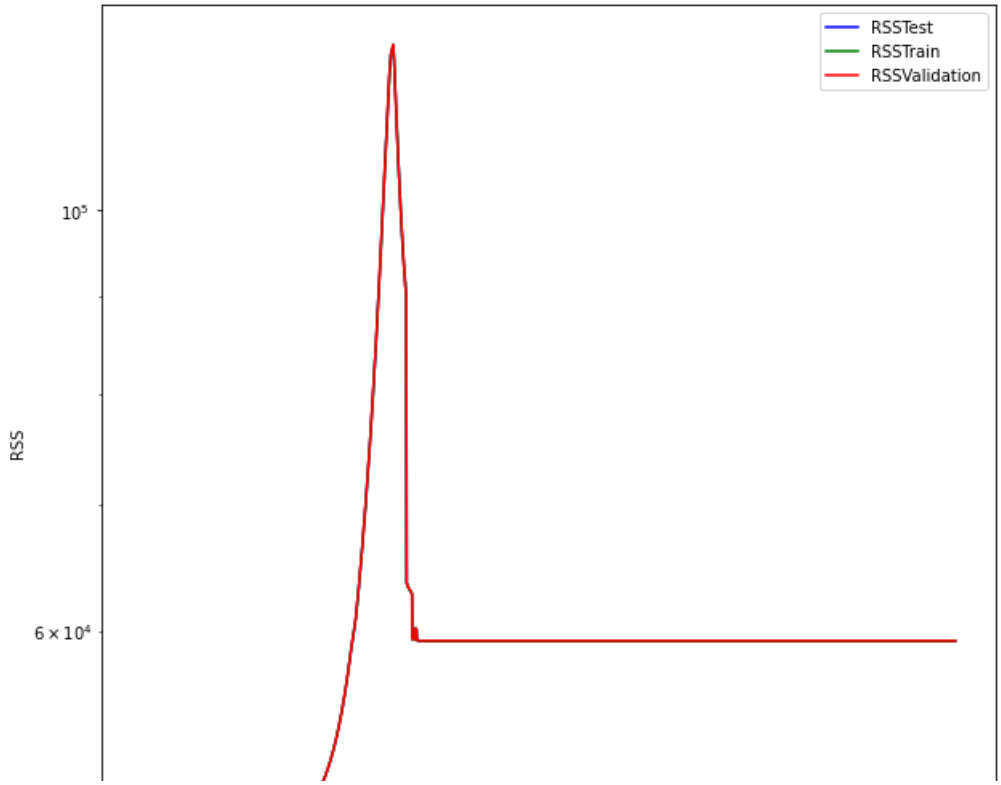
```
color=['b-','g-','r-','c-','m-','y-','k-','w-']
for i in range(len(all_features1)):

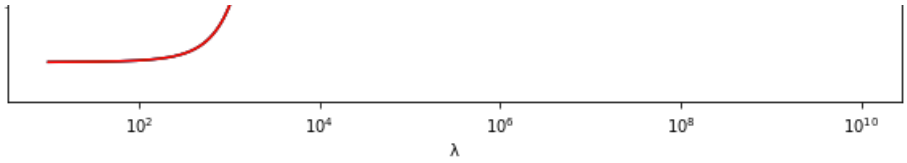
plot_coeficientes(Rl1_penalty.sort('λ'),'λ',all_features1[i],color[i%len(color)],"log","log",i,"Coe
icientes")

lRSSfeatures=['RSSTest','RSSTrain','RSSValidation']
for i in range(len(lRSSfeatures)):

plot_coeficientes(Rl1_penalty.sort('λ'),'λ',lRSSfeatures[i],color[i%len(color)],"log","log",i,"RSS"
```







Evaluación de la Predicción

Se evalua la prediccion tomando las notas de decimo de los estudiantes del año 2018 para luego comparar la prediccion con las notas de calculo de los estudiantes del año 2019.

In [214]:

```
#Rl1_penalty
#lModel[0][1].coefficients['value']#['ECONOMIA']
#len(lModel[0][1].coefficients)
def cmp(x):
    y=''
    if x<60:
       y="BAJO"
    elif x<80:
        y="BASICO"
    elif x<90:
        y="ALTO"
    else:
        y="SUPERIO"
    return y

Prediccioneval=lModel[0][1].predict(evaluation)
evaluacionre=evaluation
evaluacionre['PREDICCION']=Prediccioneval
evaluacionre['CON MINISTERIO PREDICION']=evaluacionre['PREDICCION'].apply(lambda x: cmp(x))
evaluacionre['ASERTO']=evaluacionre['cmp']==evaluacionre['CONVENCIONES MINISTERIO']
Revaluacion=evaluacionre['CALCULO','CONVENCIONES MINISTERIO','PREDICCION','CON MINISTERIO
PREDICCION','ASERTO']
Revaluacion.save('CPrediccion.csv', format='csv')
```

Cantidad de Asiertos tenidos con el modelo de prediccion.

In [215]:

```
asiertos=len(evaluacionre[evaluacionre['Aserto']==1])
Nasiertos=len(evaluacionre)-asiertos
print ("# Asiertos = "+str(asiertos)+" #No Asiertos = "+str(Nasiertos))
```

```
# Asiertos = 56 #No Asiertos =
28
```

In [216]:

```
import os
print (os.getcwd())
```

```
/home/santos/MachineRegression/Week4/Assigment1
```

In [ ]:

```

```