ITESO

Universidad Jesuita
de Guadalajara

Ciencia de datos
Proyecto 2

IF698972
Josefina Esmeralda Arriaga Hernández

10 de mayo del 2018 Guadalajara, Jalisco

Código utilizado:

```
#%%Paqueterias
import numpy as np
import pandas as pd
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.model_selection import cross_val_score
from sklearn.metrics import
confusion_matrix,accuracy_score,precision_score,recall_score,f1_score
from sklearn.preprocessing import normalize
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
import sklearn.metrics as sk
#%%Descarga de datos
data=pd.read_csv('../data/Kaggle_Training_Dataset.csv',header=0)

#%%Reporte rapido
quick_report1 = pd.DataFrame(data.describe().transpose())
quick_report2= pd.DataFrame(data.describe(include=['object']).transpose())

#%%Limpieza de datos
#se elimina la primera columna porque es el indice
data=data.drop('sku',1)

#Se elimina valores nan porque es alrededor del 10% de la info
data=data[~np.array(data.lead_time.isnull())]
#data.isnull().values.any() #comprueba si hay mas nans en dataframe

#Se cambia los valores de no/si a 0 y 1, 1(YES) siendo producto retrasado
def replace_text(x,to_replace,replacement):
    try:
        x=x.replace(to_replace,replacement)
    except:
        pass
    return x

data=data.apply(replace_text,args=('No',0))
data=data.apply(replace_text,args=('Yes',1))
#%%Seleccion de entrenamiento y prueba
X=data.iloc[:,0:21]
y=data.iloc[:,21:22]

#Normalizar datos
X=pd.DataFrame(normalize(X))

#70% entrenamiento 30% prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

```
#%%Modelos predictivos
#Logistico
lreg = LogisticRegression(random_state = 0)
lreg.fit(X_train, y_train)
y_pred = lreg.predict(X_test)
#Matriz de confusion
cm_lreg = confusion_matrix(y_test, y_pred)
#Metricas de desempeño
print('\tAccuracy: %1.3f'%accuracy_score(y_test,y_pred))
print('\tPrecision: %1.3f'%precision_score(y_test,y_pred))
print('\tRecall: %1.3f'%recall_score(y_test,y_pred))
print('\tF1: %1.3f'%f1_score(y_test,y_pred))
#Cross validation
scores_lreg=cross_val_score(lreg, X_train, y_train, cv=15)
inf_scor_lreg=pd.DataFrame([scores_lreg.mean(),scores_lreg.std()],index=['Mean','Standa
rd dev'])
#%%
#SVM
#kernel='rbf', degree=2
clf=svm.SVC(kernel='rbf', gamma=10)
clf.fit(X_train.iloc[0:100000,], y_train.iloc[0:100000,])
y_pred = clf.predict(X_test)
cm_clf = confusion_matrix(y_test, y_pred)
#Metricas de desempeño
print('\tAccuracy: %1.3f'%accuracy_score(y_test,y_pred))
print('\tPrecision: %1.3f'%precision_score(y_test,y_pred))
print('\tRecall: %1.3f'%recall_score(y_test,y_pred))
print('\tF1: %1.3f'%f1_score(y_test,y_pred))

#Cross validation
scores_clf=cross_val_score(clf, X_train.iloc[0:100000,], y_train.iloc[0:100000,], cv=5)
inf_scor_clf=pd.DataFrame([scores_clf.mean(),scores_clf.std()],index=['Mean','Standard
dev'])

#%%Eliminando variables
var=pd.DataFrame()
var['Varianza']=X.iloc[:,:].apply(lambda x: x.var())
var=var.sort_values(['Varianza'],ascending=[True])
var=(var.T)

X=X.drop([11,12,15,17,18,20],1)
X_train=X_train.drop([11,12,15,17,18,20],1)
X_test=X_test.drop([11,12,15,17,18,20],1)

var=pd.DataFrame()
var['Varianza']=X.iloc[:,:].apply(lambda x: x.var())
var=var.sort_values(['Varianza'],ascending=[True])
var=(var.T)

core=pd.DataFrame(X.corr())
```

```
X=X.drop([4,5,6,7,8,9],1)
X_train=X_train.drop([4,5,6,7,8,9],1)
X_test=X_test.drop([4,5,6,7,8,9],1)

core=pd.DataFrame(X.corr())
#%%Modelos predictivos
#Logistico
lreg = LogisticRegression(random_state = 0)
lreg.fit(X_train, y_train)
y_pred = lreg.predict(X_test)
#Matriz de confusion
cm_lreg = confusion_matrix(y_test, y_pred)
#Metricas de desempeño
print('\tAccuracy: %1.3f'%accuracy_score(y_test,y_pred))
print('\tPrecision: %1.3f'%precision_score(y_test,y_pred))
print('\tRecall: %1.3f'%recall_score(y_test,y_pred))
print('\tF1: %1.3f'%f1_score(y_test,y_pred))
#Cross validation
scores_lreg=cross_val_score(lreg, X_train, y_train, cv=15)
inf_scor_lreg=pd.DataFrame([scores_lreg.mean(),scores_lreg.std()],index=['Mean','Standa
rd dev'])
#%%
#SVM
#kernel='rbf', degree=2
clf=svm.SVC(kernel='rbf', gamma=10)
clf.fit(X_train.iloc[0:90000,], y_train.iloc[0:90000,])
y_pred = clf.predict(X_test)
cm_clf = confusion_matrix(y_test, y_pred)
#Metricas de desempeño
print('\tAccuracy: %1.3f'%accuracy_score(y_test,y_pred))
print('\tPrecision: %1.3f'%precision_score(y_test,y_pred))
print('\tRecall: %1.3f'%recall_score(y_test,y_pred))
print('\tF1: %1.3f'%f1_score(y_test,y_pred))

#Cross validation
scores_clf=cross_val_score(clf, X_train.iloc[0:90000,], y_train.iloc[0:90000,], cv=5)
inf_scor_clf=pd.DataFrame([scores_clf.mean(),scores_clf.std()],index=['Mean','Standard
dev'])

#%%Buscar el polinomio "óptimo"
#Como no se que polinomio me conviene, intento con varios, analizo y luego elijo
ngrado = 3 #Grado del polinomio
grados = np.arange(1,ngrado)
ACCURACY = np.zeros(grados.shape)
PRECISION = np.zeros(grados.shape)
RECALL = np.zeros(grados.shape)
F1 = np.zeros(grados.shape)
NUM_VARIABLES = np.zeros(grados.shape)
#%%Modelo de regresión lineal
for ngrado in grados:
    poly=PolynomialFeatures(ngrado)
```

```
    Xasterisco=poly.fit_transform(X) #es el x modificado, el que se le grega la fila de 1's
    logreg = linear_model.LogisticRegression(C=1)
    logreg.fit(Xasterisco,y) #Entrena el modelo
    Yg=logreg.predict(Xasterisco) #Sacar el "y" estimado
    #Guardar las variables en las matrices
    NUM_VARIABLES[ngrado-1] = len(logreg.coef_[0])
    ACCURACY[ngrado-1] = sk.accuracy_score(y,Yg) #Emparejamiento Simple
    PRECISION[ngrado-1] = sk.precision_score(y,Yg) #Precision
    RECALL[ngrado-1] = sk.recall_score(y,Yg) #Recall
    F1[ngrado-1] = sk.f1_score(y,Yg) #F1
#%%Visualizar los resultados
plt.plot(grados,ACCURACY)
plt.plot(grados,PRECISION)
plt.plot(grados,RECALL)
plt.plot(grados,F1)
plt.legend(('Accuracy','Precision','Recall','F1'))
plt.grid()
plt.show()
#%%Visualizar el grado de polinomio
plt.bar(grados,NUM_VARIABLES)
plt.title('Relación Grado-Parámetros')
plt.xlabel('Grado del Polinomio')
plt.ylabel('Número de Parámetros (w´s)')
plt.grid()
plt.show()
#(Por lo que se observa en las graficas la respuesta sería el poinomio de grado 4)
#%%Seleccionar el grado óptimo del análisis anterior
ngrado = 2
poly = PolynomialFeatures(ngrado)
Xasterisco = poly.fit_transform(X)
logreg = linear_model.LogisticRegression(C=1)
logreg.fit(Xasterisco,y)
Yg = logreg.predict(Xasterisco)
sk.accuracy_score(y,Yg) #Porcentaje de acierto en total, y lo muestra en la terminal
#%% Anlaisiar los coeficientes  más significativos
W = logreg.coef_[0]
plt.bar(np.arange(len(W)),W)
plt.title('Relación Varaible-Valor del Parametro')
plt.xlabel('Número de Varible (x´s)')
plt.ylabel('Valor del Parámetro (w´s)')
plt.show()
#%%Anlaizar los coeficientes más significativos
W = logreg.coef_[0]
Wabs = np.abs(W)
umbral = 0.5 #umbral que indica que tan significante o insignificante es el valor de un
parámetro
indx = Wabs>umbral
Xasterisco_seleccionada = Xasterisco[:,indx] #Sub matriz de x asterisco con las variables
de los parametros significativos
plt.bar(np.arange(len(W[indx])),W[indx])
plt.title('Relación Varaible-Valor del Parametro Significativos')
```

```
plt.xlabel('Número de Varible (x´s)')
plt.ylabel('Valor del Parámetro (w´s)')
plt.show()
#%%Reentrenar el modelo con las variables seleccionadas
logreg_entrenada = linear_model.LogisticRegression(C=1)
logreg_entrenada.fit(Xasterisco_seleccionada,y)
Yg_entrenado = logreg_entrenada.predict(Xasterisco_seleccionada)
sk.accuracy_score(y,Yg_entrenado) #Porcentaje de acierto en total, y lo muestra en la
terminal
diferencia = sk.accuracy_score(y,Yg) - sk.accuracy_score(y,Yg_entrenado)
print('la diferencia en porcentaje de aciertos del modelo entrenado y no entrenado es: ')
print(diferencia)
#Se observa que pese a tener menos variables, el porcentaje de accuracy score
entrenado
#y el porcentahe de acierto sin entrenar, es el mismo. Es decir que con menos variables
#se llegó exactamente al mismo resultado. (con umbral de 0.5)
#se hace las metricasde desemepeño
print('\tAccuracy: %1.3f'%accuracy_score(y_test,y_pred))
print('\tPrecision: %1.3f'%precision_score(y_test,y_pred))
print('\tRecall: %1.3f'%recall_score(y_test,y_pred))
print('\tF1: %1.3f'%f1_score(y_test,y_pred))
```

---

1. Realizar un estudio de calidad de los datos.

Como primer paso se descarga la información y se realiza un análisis rápido de los datos:

Index	count	mean	std	min	25%	50%	75%	max
sku	1.69305e+06	3.44246e+06	775159	2.06362e+06	2.98351e+06	3.40677e+06	3.83004e+06	4.98646e+06
national_inv	1.69305e+06	494.057	29229.4	-33423	4	15	81	1.22851e+07
lead_time	1.59172e+06	7.90869	7.04047	0	4	8	9	52
in_transit_...	1.69305e+06	40.3174	1066.34	0	0	0	0	489408
forecast_3_...	1.69305e+06	181.677	5458.87	0	0	0	4	1.51116e+06
forecast_6_...	1.69305e+06	343.122	9703.41	0	0	0	12	2.21875e+06
forecast_9_...	1.69305e+06	501.845	13825.7	0	0	0	20	3.22929e+06
sales_1_mon...	1.69305e+06	53.6559	1689.28	0	0	0	4	741762
sales_3_mon...	1.69305e+06	173.569	5151.45	0	0	1	14	1.19241e+06
sales_6_mon...	1.69305e+06	340.796	9399.14	0	0	2	31	2.13356e+06
sales_9_mon...	1.69305e+06	512.029	13990	0	0	4	47	3.20517e+06
min_bank	1.69305e+06	52.3712	1281.74	0	0	0	3	366462
pieces_past...	1.69305e+06	1.73745	261.532	0	0	0	0	162332
perf_6_mont...	1.69305e+06	-6.8916	26.5709	-99	0.61	0.82	0.96	1
perf_12_mon...	1.69305e+06	-6.43768	25.8393	-99	0.66	0.81	0.95	1
local_bo_qty	1.69305e+06	0.65072	42.1517	0	0	0	0	15000

Se muestra que lead time no tiene valores en algunas filas por lo que será necesario eliminar los valores nan, el promedio de las columnas va desde 25 hasta 775159, y los valores máximos de las columnas van desde 1 hasta 12285100.

Index	count	unique	top	freq
potential_i...	1693050	2	No	1692237
deck_risk	1693050	2	No	1347759
oe_constrai...	1693050	2	No	1692744
ppap_risk	1693050	2	No	1491577
stop_auto_b...	1693050	2	Yes	1627337
rev_stop	1693050	2	No	1692340
went_on_bac...	1693050	2	No	1682136

En los datos cualitativos se observa que son valores Yes o No por lo que se podrá pasar a numérico donde No será 0 y Yes 1 para poder hacer una predicción cuando los productos entran en retraso.

2. Seleccione una muestra de los datos, como datos de entrenamiento y los datos de prueba o “crossvalidation”. La forma de selección recomendada es de forma aleatoria cuidando que la proporción de datos de cada categoría se mantenga como la base de datos original.

Antes de hacer la selección de muestra de datos se realiza una limpieza en donde se elimina la primera columna porque son los Ids de las filas y se elimina los valores NaN porque hay 1693050 y cuando se eliminan se mantiene 1591716 datos esto quiere decir que los valores nan solo representan 5% de los datos. También se cambian las columnas en donde dice Yes/No a numérico siendo 0=No y 1=Si, esto con la finalidad que se haga la predicción de los productos que si entran en backorder o se retrasan.

oe_constraint	ppap_risk	stop_auto_buy	rev_stop	went_on_backorder
No	No	Yes	No	No
No	No	Yes	No	No
No	No	Yes	No	No
No	No	Yes	No	No
No	No	No	No	No
No	No	Yes	No	No
No	No	Yes	No	No

oe_constraint	ppap_risk	stop_auto_buy	rev_stop	went_on_backorder
0	1	1	0	0
0	0	1	0	0
0	0	0	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0

Como las filas no son una serie de tiempo se eligen de manera aleatoria en donde 70% de los datos seran de entrenamiento y el 30% serán de prueba con la función train_test_split de sklearn. Antes de hacer el split se separa las variables en las variables predictoras y la variable a predecir, en donde la variable a predecir es la última columna went_on_backorder y las demás columnas son con las que se va hacer la predección. Al ver el análisis de datos se observa que hay valores de rangos muy diferentes por lo que se normaliza la variable X y después ya se hacen las separaciones de entrenamiento y prueba.

3. Diseñar un modelo logístico y una maquina de vectores soporte (SVM) utilizando todas las variables de la base de datos para clasificar los registros.

Se utiliza la regresión logística y SVM, no se mueve ningún hiperparámetro de la regresión logística y de SVM se usa “Radial Basis Function” y gamma de 10, en donde se realiza los métricos de desempeño de cada uno, su matriz de confusión y una validación cruzada de ambos modelos.

Para la regresión logística se obtiene los siguientes scores y matriz de confusión:

```
Accuracy: 0.993
Precision: 0.000
Recall: 0.000
F1: 0.000
```

	0	1
0	474326	31
1	3158	0

Se observa un buen comportamiento en el accuracy score pero las demás métricas muestran un comportamiento malo, al hacer el cross validation también se observa que hay un buen score y tiene una desviación muy pequeña.

Index	0
Mean	0.993267
Standard dev	3.14077e-05

Para el modelo SVM se obtiene los siguiente scores y matriz de confusión:

```
Accuracy: 0.993
Precision: 0.154
Recall: 0.001
F1: 0.001
```

	0	1
0	474346	11
1	3156	2

Se observa un buen comportamiento en el accuracy score pero las demás métricas muestran un comportamiento medio, al hacer el cross validation también se observa que hay un buen score y tiene una desviación muy pequeña.

Index	0
Mean	0.99387
Standard dev	2.42453e-05

Por lo que ambos modelos tienen el mismo comportamiento, que se puede optimizar.

4. Determinar basado en los criterios de selección de variables (varianza, correlación, PCA), cuáles son las variables que posiblemente aporten información al modelo y/o no sean importantes. (Selección pre-modelado)

Primero se observa el comportamiento entre las variables X con varianza y correlación para eliminar los que tengan una variación mínima que no afecte el modelo y una correlación muy grande donde si se elimina uno de los dos no se va a afectar el modelo.

Index	Varianza	Index	Varianza
17	3.03694e-07	2	0.00609565
20	1.90682e-06	13	0.00837961
11	2.64167e-06	7	0.00917452
15	4.01363e-05	3	0.0161392
12	0.000134694	8	0.030936
18	0.000659848	4	0.032943
16	0.00113961	5	0.05973
6	0.00169511	9	0.067817
19	0.00300572	1	0.119166
10	0.00428369	0	0.120803
14	0.00598971		

Se oberva que los que tienen menos variación son las variables oe_constraint (17), rev_stop (20), potential_issue (11), local_bo_qty (15), pieces_past_due (12), ppap_risk(18), por lo que se eliminen de las variables X_train y X_test. El siguiente paso es checar la matriz de correlación:

Index	0	1	2	3	4	5	6	7	8	9	10	13	14	16	19
0	1	-0.194…	-0.163…	-0.531…	-0.60…	-0.61…	-0.27…	-0.382…	-0.396…	-0.384…	-0.12…	0.0460…	0.0312…	0.0572…	-0.050…
1	-0.1946…	1	-0.130…	-0.304…	-0.40…	-0.44…	-0.31…	-0.440…	-0.473…	-0.473…	-0.07…	0.3713…	0.4193…	0.1503…	0.6809…
2	-0.1638…	-0.130…	1	0.0547…	0.098…	0.113…	0.097…	0.1420…	0.1364…	0.1244…	0.056…	-0.021…	-0.032…	0.0478…	-0.077…
3	-0.5314…	-0.304…	0.0547…	1	0.853…	0.744…	0.178…	0.2403…	0.21467	0.1984…	0.046…	-0.070…	-0.098…	0.0018…	-0.225…
4	-0.6088…	-0.406…	0.0988…	0.8535…	1	0.927…	0.274…	0.3992…	0.3992…	0.3853…	0.091…	-0.093…	-0.130…	-0.065…	-0.318…
5	-0.6162…	-0.442…	0.11386	0.7446…	0.927…	1	0.328…	0.4933…	0.5090…	0.4945…	0.122…	-0.103…	-0.144…	-0.116…	-0.360…
6	-0.27816	-0.316…	0.0973…	0.1785…	0.274…	0.328…	1	0.7433…	0.6326…	0.5824…	0.143…	-0.063…	-0.091…	-0.100…	-0.2547
7	-0.3826…	-0.440…	0.1420…	0.2403…	0.399…	0.493…	0.743…	1	0.8791	0.8193…	0.199…	-0.092…	-0.132…	-0.161…	-0.368…
8	-0.3961…	-0.473…	0.1364…	0.21467	0.399…	0.509…	0.632…	0.8791	1	0.9478…	0.228…	-0.098…	-0.142…	-0.197…	-0.405…
9	-0.3840…	-0.473…	0.1244…	0.1984…	0.385…	0.494…	0.582…	0.8193…	0.9478…	1	0.244…	-0.100…	-0.144…	-0.211…	-0.413…
10	-0.1289…	-0.076…	0.0560…	0.0460…	0.091…	0.122…	0.143…	0.1996…	0.22822	0.24438	1	0.0087…	-0.003…	-0.014…	-0.072…
13	0.04606…	0.3713…	-0.021…	-0.070…	-0.09…	-0.10…	-0.06…	-0.092…	-0.098…	-0.100…	0.008…	1	0.85281	0.1672…	0.4938…
14	0.03121…	0.4193…	-0.032…	-0.098…	-0.13…	-0.14…	-0.09…	-0.132…	-0.142…	-0.144…	-0.00…	0.85281	1	0.1880…	0.5626…
16	0.05722…	0.1503…	0.0478…	0.0018…	-0.06…	-0.11…	-0.10…	-0.161…	-0.197…	-0.211…	-0.01…	0.1672…	0.1880…	1	0.3860…
19	-0.0503…	0.6809…	-0.077…	-0.225…	-0.31…	-0.36…	-0.25…	-0.368…	-0.405…	-0.413…	-0.07…	0.4938…	0.5626…	0.3860…	1

Se observa que las variables que tienen mas correlación con las demás son forecast_6_month (4), forecast_9_month (5), sales_1_month (6), sales_3_month (7), sales_6_month (8), sales_9_month (9). Se tiene valores menores a cero por lo que no es posible hacer la reducción PCA, de igual manera se considera que ya se eliminaron suficientes variables para que se haga un modelo más acertado.

5. Diseñar un modelo logístico y un modelo SVM con las variables resultantes del análisis de selección de variables.

Se realiza otravez el modelo con las misma división de entrenamiento y prueba con la nueva reducción de variables.

Para la regresión logística se obtiene los siguientes scores y matriz de confusión:

	0	1
0	474343	14
1	3158	0

```
Accuracy: 0.993
Precision: 0.056
Recall: 0.000
F1: 0.001
```

Se observa un buen comportamiento en el accuracy score, al hacer el cross validation también se observa que hay un buen score y tiene una desviación muy pequeña. Muestra un mejor comportamiento en las métrias de desempeño pero no ideales.

Index	0
Mean	0.993297
Standard dev	2.27689e-05

Para el modelo SVM se obtiene los siguiente scores y matriz de confusión:

	0	1
0	474347	10
1	3158	0

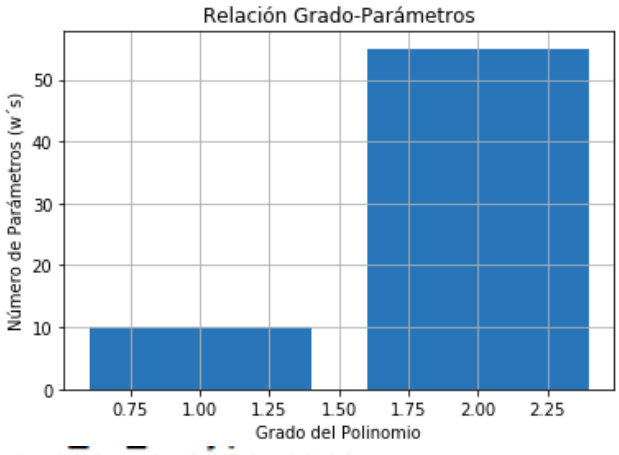
```
Accuracy: 0.993
Precision: 0.000
Recall: 0.000
F1: 0.000
```

Se observa un buen comportamiento en el accuracy score, al hacer el cross validation también se observa que hay un buen score y tiene una desviación muy pequeña. Comparandolo con la regresión logística es mejor el primero modelo ya que el SVM no demuestra mejorías después de la limpieza antes del modelado.

Por lo que ambos modelos tienen el mismo comportamiento, que se puede optimizar.

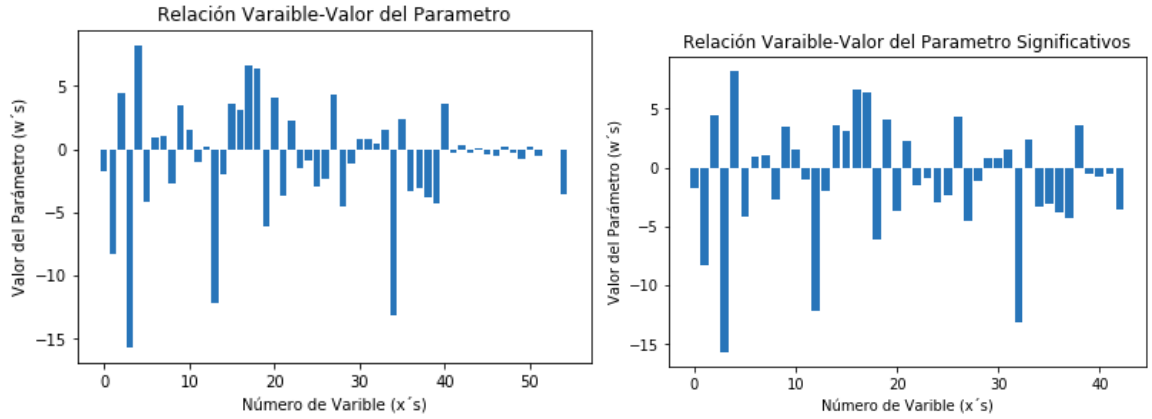
6. Solo del modelo logístico, realizar un análisis/verificación de "overfitting" y revisar si el modelo puede ser reducido, tomando en cuenta la magnitud de los parámetros del modelo resultantes. (Selección post-modelado)

Para reducir el modelo se utiliza la función de polinomios para encontrar el ideal para la regresión logística, en este caso el mejor resultado es grado 2 porque tiene una accuracy de 99%.




```
0.9931507882059363
```

De igual manera se eliminan las variables menos importantes, con un umbral de 0.5 pero al verificar si el porcentaje de aciertos es mejor o peor que el nuevo modelo se comprueba que no hay una diferencia significativa.




```
la diferencia en porcentaje de aciertos del modelo entrenado y no entrenado
es:
5.968401398248613e-05
```

Se obtiene de nuevo la matriz de confusión y sus scores:

	0	1
0	1580696	447
1	10550	23

Accuracy: 0.993
Precision: 0.049
Recall: 0.002
F1: 0.004

7. Conclusiones

En conclusión el mejor modelo es la regresión logística cuando se optimiza antes y después del modelado, otra opción sería usar otro modelo que se pueda cambiar más los hiperparámetros y que sea un modelo más exacto sin llegar a overfitting, en el caso de SVM no es el mejor modelo ya que tarda mucho para hacer las predicciones, ocupando mucha memoria y sus resultados no son mejores que regresión logística que tarda menos en obtener las predicciones.