



Tema 3 : Introducción a Python
y a los paradigmas de datos

IPPPD: Entrega ejercicios Kaggle (Cs570 y AllState)

Inmaculada Perea Fernández

Índice

Introducción	2
CS570 - Descripción del problema.....	4
CS570 - Inspección de los datos.....	5
CS570 - Configuración del entorno.....	8
CS570 – Selección de modelos y entrenamiento	9
CS570 - Resultados.....	10
Submissions de Kaggle	10
Ranking de Kaggle	11
AllState - Descripción del problema	13
AllState - Inspección de los datos.....	14
AllState - Configuración del entorno.....	14
AllState – Selección de modelos y entrenamiento	15
AllState - Resultados.....	16
Submission de Kaggle.....	17
Ranking de Kaggle	18
Referencias.....	19

Introducción

En el siguiente documento se realizará un resumen del trabajo realizado para resolver los ejercicios propuestos para la evaluación del tema 3 de la asignatura IPPD.

Se detallan las acciones llevadas a cabo para encontrar la solución óptima a las competiciones de Kaggle propuestas. Algunas de las acciones más destacables son: la inspección de los datos, la configuración y preparación del entorno, las transformaciones de variables, la selección y entrenamiento de modelos y la optimización de parámetros.

Se ha utilizado el framework de machine learning estudiado en clase, configurándolo convenientemente e implementando las clases necesarias para resolver los ejercicios.

Por último se mostrarán los resultados obtenidos y la puntuación privada proporcionada por Kaggle.

El documento se divide en dos secciones distintas, una para la competición CS570 y otra para la competición AllState.

CS570

(<https://inclass.kaggle.com/c/newcs570midtermchallenge>)

CS570 - Descripción del problema

A continuación se muestra la descripción y los criterios de evaluación del problema que se plantea en la competición CS570 [1].

Completed • Knowledge • 43 teams

New CS570 Midterm Challenge

Mon 10 Oct 2016 – Mon 5 Dec 2016 (31 days ago)

[Competition Details](#) » [Get the Data](#) » [Make a submission](#)

Customer Retention Classifier

This will be CS 570's midterm assignment. For the midterm, we will be using everything you've learned so far to build a customer "churn" model.

Imagine you work for GreenPlate, the west coast's fastest growing new subscription box service. GreenPlate's customers get a monthly box, full of healthy and sometimes delicious snacks. GreenPlate wants to know which customers are most likely to cancel their subscription service so that they can use incentives to retain those customers.

You need to use the training dataset given to create a model that predicts if the customers in the testing dataset will cancel their subscription to GreenPlate.

The dataset contains multiple features, and you can use any of those features you want. The names of those features have been hidden from you.

The dependent variable is y . If $y=0$ that means the customer didn't cancel. If $y=1$ that means the customer DID cancel. You should predict the probability of cancellation because your model will be evaluated using AUC.

You have two separate deliverables, listed below:

- 1. Please submit your answer via Kaggle. You'll need to be on the leaderboard to receive credit for this assignment.**
- 2. Please submit all the code you used for cleaning, prepping, and modeling your data via blackboard (NOT github).**

Evaluation

Evaluation

The evaluation metric that this competition will be using is Area Under the Curve (AUC).

Submission Format

For every entry in the test set, submission files should contain two columns: Id and Predicted Probability of cancellation.

The file should contain a header and have the following format:

```
id,y
1,0
2,.1
3,.9
4,.4
etc.
```

Grading

Your midterm will be scored by:

- 50% Your model implementation
- 25% Your data cleaning/prep code
- 25% Your leaderboard score vs your classmates.

CS570 - Inspección de los datos

El primer lugar inspeccionamos los datos con el propósito de localizar datos inexistentes, incompletos, erróneos, etc. Esto nos dará idea del pre-procesado necesario antes de comenzar con la selección de modelos y el entrenamiento de los mismos.

Para este cometido he creado un fichero notebook ipython [3] que utiliza las librerías pandas y numpy para explorar los ficheros csv de training y determinar el tipo de datos de las variables a analizar, la existencia de valores NaN, infinitos, variables categóricas, cadenas de texto a limpiar, etc.

```
In [3]: # Inspecciono los 5 primeros elementos
train_cs570.head()
```

```
Out[3]:
```

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	...	x41	x42	x43	x
0	-0.166563	-3.961588	4.621113	2.481908	-1.800135	0.804684	6.718751	-14.789997	-1.040673	-4.204950	...	-1.497117	5.414063	-2.325655	1
1	-0.149894	-0.585676	27.839856	4.152333	6.426802	-2.426943	40.477058	-6.725709	0.896421	0.330165	...	36.292790	4.490915	0.762561	6
2	-0.321707	-1.429819	12.251561	6.586874	-5.304647	-11.311090	17.812850	11.060572	5.325880	-2.632984	...	-0.368491	9.088864	-0.689886	-1
3	-0.245594	5.076677	-24.149632	3.637307	6.505811	2.290224	-35.111751	-18.913592	-0.337041	-5.568076	...	15.691546	-7.467775	2.940789	-1
4	-0.273366	0.306326	-11.352593	1.676758	2.928441	-0.616824	-16.505817	27.532281	1.199715	-4.309105	...	-13.911297	-5.229937	1.783928	3

5 rows x 51 columns

```
In [4]: train_cs570.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 160000 entries, 0 to 159999
Data columns (total 51 columns):
x0      159974 non-null float64
x1      159975 non-null float64
x2      159962 non-null float64
x3      159963 non-null float64
x4      159974 non-null float64
...
x46     159969 non-null float64
x47     159963 non-null float64
x48     159968 non-null float64
x49     159968 non-null float64
y       160000 non-null int64
dtypes: float64(45), int64(1), object(5)
memory usage: 62.3+ MB
```

```
In [41]: # Comprueba la existencia de NaN
True in np.isnan(train_cs570.as_matrix(columns=[train_cs570.columns[0]]))
```

```
Out[41]: True
```

```
In [40]: # Comprueba la existencia de infinitos
True in np.isinf(train_cs570.as_matrix(columns=[train_cs570.columns[0]]))
```

```
Out[40]: False
```

Observamos que los datos contienen NaN, con lo cual en el pre-procesado a realizar antes de la selección de modelos será necesario limpiarlos. Para ello utilizaremos la clase *ImputerTransform* existente en *Feature_transform.py*

```

64 class ImputerTransform(BaseEstimator, TransformerMixin):
65
66     def __init__(self, missing_values='NaN', strategy=-999999):
67         self.missing_values = missing_values
68         self.strategy = strategy
69
70
71     def fit(self, X, y=None):
72         return self
73
74     def transform(self, X):
75
76         if self.strategy in ['mean', 'median', 'most_frequent']:
77
78             imp = Imputer(self.missing_values, self.strategy)
79             X = imp.fit_transform(X)
80
81         # los algoritmos siempre trabajan con numeros, hay que sustituir los caracteres
82         else:
83             for c in X.columns:
84
85                 if X[c].dtype == np.object:
86                     strategy = str(self.strategy)
87                 else:
88                     strategy = int(self.strategy)
89
90                 if self.missing_values == "NaN":
91                     X.loc[pd.isnull(X[c].values), c] = strategy
92                 else:
93                     X.loc[X[c].isin([self.missing_values]), c] = strategy
94
95         return X

```

A continuación inspeccionaremos las 5 variables tipo “object” para determinar qué procesamiento será necesario realizar. Encontramos que existen tres variables (x24, x29 y x30) categóricas que habrá que codificar correctamente para trabajar con los modelos.

```
In [18]: train_cs570["x24"].head()
```

```
Out[18]: 0    euorpe
1     asia
2     asia
3     asia
4     asia
Name: x24, dtype: object
```

```
In [15]: train_cs570["x29"].head()
```

```
Out[15]: 0    July
1    Aug
2    July
3    July
4    July
Name: x29, dtype: object
```

```
In [17]: train_cs570["x30"].head()
```

```
Out[17]: 0    tuesday
1   wednesday
2   wednesday
3   wednesday
4    tuesday
Name: x30, dtype: object
```

Para el tratamiento de estas variables he implementado una nueva clase *Encoder* en *Feature_transform.py*, esta clase está basada en un encoder utilizado en la versión 1 del framework, pero he tenido que adaptarla ya que en este caso el tipo de dato a reemplazar es de tipo ‘object’ y no ‘str’. Esta clase utiliza el método *LabelEncoder* de *sklearn* para codificar las variables categóricas.

```

19 class Encoder(BaseEstimator, TransformerMixin):
20
21     def __init__(self, columns=None):
22         self.columns = columns
23
24     def fit(self, X, y):
25         return self
26
27     def transform(self, X):
28         columns = self.columns
29         if not columns:
30             columns = X.columns
31
32         for c in columns:
33             if (X[c].dtype == 'object'):
34                 X[c] = LabelEncoder().fit_transform(X[c])
35         return X

```

Por último nos queda por inspeccionar las variables x32 y x37, en las que podemos observar que los datos incluyen el carácter % y \$ junto con datos de tipo numérico.

```
In [13]: train_cs570["x32"].head()
```

```

Out[13]: 0    0.0%
1   -0.02%
2   -0.01%
3    0.01%
4    0.01%
Name: x32, dtype: object

```

```
In [14]: train_cs570["x37"].head()
```

```

Out[14]: 0    $1313.96
1    $1962.78
2    $430.47
3    $-2366.29
4    $-620.66
Name: x37, dtype: object

```

En este caso utilizaremos la clase RemoveStringTransform existente en el framework, que reemplaza los caracteres que se le pasan como argumento por una cadena vacía.

```

37
38 class RemoveStringsTransform(BaseEstimator, TransformerMixin):
39
40     def __init__(self, strings, columns=None):
41         self.strings = strings
42         self.columns = columns
43
44
45     def fit(self, X, y):
46         return self
47
48
49     def transform(self, X):
50         # Si no se especifica nada se aplica a todas las columnas
51         columns = self.columns
52         if not columns:
53             columns = X.columns
54
55         for c in columns:
56             for string in self.strings:
57                 # Compruebo si tiene caracter, y si lo tiene lo sustituyo para cadena vacia
58                 if X[c].dtype == np.object and X[c].str.contains(string).any():
59                     X[c] = X[c].str.replace(string, '')
60
61         return X
62

```


CS570 - Configuración del entorno

En este apartado detallaré los principales parámetros configurados en el fichero CS570/config.py [5] utilizado para el pre-procesado, selección de modelos y generación de predicciones con el framework.

He realizado varias pruebas de pre-procesado para este ejercicio, y el que se muestra a continuación es el que me ha ofrecido mejores resultados.

```
49 PREPROCESSING = [  
50  
51     ('scaler_int', [  
52         ('dropNaN', ImputerTransform()),  
53         ('clean', RemoveStringsTransform(strings=['%', '$'])),  
54         ('encode', Encoder()),  
55         # ('log', LogTransform()),  
56         ('inter', InteractionTransform(  
57             interactions=['add'],  
58             columns=['x%d' % i for i in range(1, 10)])),  
59         ('std', StandardScaler()),  
60         ('fs', TreeBased('extra_trees_regressor', 20, 190)),  
61     ]),  
62 ],  
63
```

He utilizado métrica log_loss para calcular el score y comparar los distintos modelos generados

```
39  
40 SCORER = make_scorer(log_loss,  
41                       needs_proba=True,  
42                       greater_is_better=False)  
43
```

También he cambiado el parámetro TARGET porque en este ejercicio la variable objetivo es la "y".

```
TARGET = "y"  
TARGET_TRANSFORM = LabelTransform
```

Para crear las predicciones tal y como las requiere kaggle tuve que utilizar la clase DummyId para que generase la columna con el Id en el csv de submission.

```
31  
32 PREDICTIONS_DIR = 'cs570/submissions'  
33 PREDICTIONS_PREPROCESS = [DummyId()]  
34 PREDICTION_COLUMNS = ['id']
```

A continuación muestro una captura de todos los modelos con los que he trabajado y que he configurado convenientemente en el fichero config. El modelo KNN no lo he generado para este ejercicio puesto que tarda mucho en procesar (más de 3 horas) y he abortado su generación para poder realizar más pruebas con otros modelos que no necesitan tanto tiempo de procesado.

```

67 MODELS = {
68     ('lr', LogisticRegression(fit_intercept=True, solver='newton-cg',
69                               multi_class='multinomial')),
70     ('rf', RandomForestClassifier(random_state=RANDOM_STATE)),
71     ('xgbClassifier', XGBClassifier(base_score=0.5, colsample_bylevel=0.6, colsample_bytree=0.6,
72                                   gamma=0, learning_rate=0.1, max_delta_step=0,
73                                   max_depth=10, min_child_weight=1,
74                                   n_estimators=1000, nthread=-1, objective='binary:logistic',
75                                   reg_alpha=0.6, reg_lambda=1, scale_pos_weight=1, silent=True,
76                                   subsample=1)),
77
78     ('xgb', XGBRegressor()),
79     ('lasso', Lasso(fit_intercept=True)), # inma
80     ('ridge', Ridge(fit_intercept=True)),
81     ('elastic_net', ElasticNet(fit_intercept=True)),
82     ('bayes_ridge', BayesianRidge(fit_intercept=True)),
83     ('sgd', SGDRegressor()),
84     ('gb', GradientBoostingRegressor()),
85     #('KNN', KNeighborsRegressor(n_neighbors=10)),

```

CS570 – Selección de modelos y entrenamiento

Una vez obtuve resultados con distintos modelos de clasificación y regresión logística probé a realizar transformaciones de variables y optimización de parámetros para mejorar los resultados:

```

Project: cs570
Date: Sun Nov 27 17:29:45 2016
Random state: 2016
Target transform: LabelTransform
Parameter optimization: True
Train data shape: 160000, 52

```

Preprocess	Model	Score	STD
scaler_int	rf	0.33567	0.00200

```

Project: cs570
Date: Sun Nov 27 18:13:32 2016
Random state: 2016
Target transform: LabelTransform
Parameter optimization: True
Train data shape: 160000, 52

```

Preprocess	Model	Score	STD
scaler_int	lr	0.67800	0.00011

```

Project: cs570
Date: Tue Jan 3 22:23:26 2017
Random state: 2016
Target transform: LabelTransform
Parameter optimization: True
Train data shape: 160000, 52

```

Preprocess	Model	Score	STD
scaler_int	ridge	0.67379	0.00015

```

Project: cs570
Date: Tue Jan 3 22:27:43 2017
Random state: 2016
Target transform: LabelTransform
Parameter optimization: True
Train data shape: 160000, 52

```

Preprocess	Model	Score	STD
scaler_int	elastic_net	0.67351	0.00000

- 1) Probé a realizar el logaritmo usando la clase LogTransform, pero obtuve errores, creo que se debe a que hay números demasiado grandes, así que abandoné ese camino.
- 2) Probé a realizar operaciones entre columnas (suma y multiplicación) con la clase InteractionTransform, pero no obtuve mejora del score. Probé primero con las 10 primeras variables (x_1, \dots, x_{10}) y luego con 20 variables (x_1, \dots, x_{20}) sin resultados positivos.
- 3) Los que mejores resultados los obtuve con los modelos rf y gb, así que me centré en ellos e hice varias pruebas variando algunos de los META_PARAMETERS: n_estimators, max_features y max_depth pero tampoco he conseguido mejoras significativas.

- 4) También he probado a variar los parámetros `learning_rate`, `max_depth` y `n_estimators` del modelo `xgboostClassifier`, pero no he conseguido mejorar los resultados obtenidos con `RandomForestClassifier`
- 5) No he podido probar KNN porque el tiempo que necesita para finalizar es muy alto y me he visto obligada abortar la ejecución en 2 ocasiones.

CS570 - Resultados

El modelo con el que mejores resultados he obtenido es con *RandomForestClassifier*

```
Project: cs570
Date: Sun Nov 27 17:29:45 2016
Random state: 2016
Target transform: LabelTransform
Parameter optimization: True
Train data shape: 160000, 52
```

Preprocess	Model	Score	STD
scaler_int	rf	0.33567	0.00200

Con los siguientes parámetros:

- `min_samples_split=1`,
- `n_estimators=20`,
- `criterion=gini`
- `min_samples_leaf=3`
- `bootstrap=True`
- `min_impurity_split=1e-07`
- `max_features=auto`
- `n_jobs=1`
- `random_state=2016`

Submissions de Kaggle

A continuación los 3 modelos con los que he obtenido los mejores resultados en kaggle.

New CS570 Midterm Challenge

Mon 10 Oct 2016 – Mon 5 Dec 2016 (31 days ago)

Your Submissions

You are submitting as part of team **inma perea**.

[Make a submission »](#)

The competition deadline has already passed and you can no longer modify selections. While this competition was active, you could select up to 2 submissions. This information is provided for historical purposes only.

Submission	Files	Public Score	Private Score	Selected?
Post-Deadline: Sun, 18 Dec 2016 12:38:49 Edit description	rf_0.33567.csv	0.95448	0.95705	<input type="checkbox"/>
Post-Deadline: Wed, 04 Jan 2017 20:35:10 Edit description	gb_0.44365.csv	0.90490	0.90843	<input type="checkbox"/>
Post-Deadline: Thu, 05 Jan 2017 18:43:50 Edit description	gb_0.44365.csv	0.90490	0.90843	<input type="checkbox"/>
Post-Deadline: Wed, 04 Jan 2017 20:38:40 Edit description	bayes_ridge_0.67356.csv	0.50173	0.50005	<input type="checkbox"/>

Ranking de Kaggle

23	—	AlvaroSanchez	0.96994	3	Mon, 05 Dec 2016 13:30:02
24	—	gamgee	0.96327	2	Sun, 30 Oct 2016 19:28:45
-		inma perea	0.95705	-	Fri, 06 Jan 2017 18:30:56 Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
25	↑2	AratiYadav	0.94227	2	Mon, 24 Oct 2016 22:13:01
26	↓1	AbhijeetUpadhyay	0.94138	7	Mon, 05 Dec 2016 20:22:01 (-23.1h)

Allstate

(<https://www.kaggle.com/c/allstateclaimsseverity>)

AllState - Descripción del problema

A continuación se muestra la descripción y el modo de evaluación del problema que se plantea en la competición AllState [2].

Completed • Jobs • 3,055 teams

Allstate Claims Severity

Mon 10 Oct 2016 – Mon 12 Dec 2016 (24 days ago)

[Competition Details](#) » [Get the Data](#) » [Make a submission](#)

How severe is an insurance claim?

When you've been devastated by a serious car accident, your focus is on the things that matter the most: family, friends, and other loved ones. Pushing paper with your insurance agent is the last place you want your time or mental energy spent. This is why Allstate, a personal insurer in the United States, is continually seeking fresh ideas to improve their claims service for the over 16 million households they protect.



Allstate is currently developing automated methods of predicting the cost, and hence severity, of claims. In this recruitment challenge, Kagglers are invited to show off their creativity and flex their technical chops by creating an algorithm which accurately predicts claims severity. Aspiring competitors will demonstrate insight into better ways to predict claims severity for the chance to be part of Allstate's efforts to ensure a worry-free customer experience.

New to Kaggle? This competition is a recruiting competition, your chance to get a foot in the door with the hiring team at Allstate.

Evaluation

Submissions are evaluated on the [mean absolute error \(MAE\)](#) between the predicted loss and the actual loss.

Submission File

For every id in the test set, you should predict the loss value. The file should contain a header and have the following format:

```
id,loss
4,0
6,1
9,99.3
etc.
```

AllState - Inspección de los datos

Del mismo modo que en el ejercicio anterior inspeccionamos los datos para determinar el pre-procesado antes de empezar con la selección de modelos.

```
In [5]: train_allstate.head()
```

Out[5]:

	id	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	cont14	loss
0	1	A	B	A	B	A	A	A	A	B	...	0.718367	0.335060	0.30260	0.67135	0.83510	0.569745	0.594646	0.822493	0.714843	2213.1
1	2	A	B	A	A	A	A	A	A	B	...	0.438917	0.436585	0.60087	0.35127	0.43919	0.338312	0.366307	0.611431	0.304496	1283.1
2	5	A	B	A	A	B	A	A	A	B	...	0.289648	0.315545	0.27320	0.26076	0.32446	0.381398	0.373424	0.195709	0.774425	3005.1
3	10	B	B	A	B	A	A	A	A	B	...	0.440945	0.391128	0.31796	0.32128	0.44467	0.327915	0.321570	0.605077	0.602642	939.8
4	11	A	B	A	B	A	A	A	A	B	...	0.178193	0.247408	0.24564	0.22089	0.21230	0.204687	0.202213	0.246011	0.432606	2763.1

5 rows x 132 columns

```
In [6]: train_allstate.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188318 entries, 0 to 188317
Columns: 132 entries, id to loss
dtypes: float64(15), int64(1), object(116)
memory usage: 189.7+ MB
```

```
In [39]: # Comprueba La existencia de NAN
True in np.isnan(train_allstate.as_matrix(columns=[train_allstate.columns[0]]))
```

Out[39]: False

```
In [38]: # Comprueba La existencia de infinitos
True in np.isinf(train_allstate.as_matrix(columns=[train_allstate.columns[0]]))
```

Out[38]: False

Al contrario que en el ejercicio anterior en este ejercicio no es necesario realizar limpieza de datos ya que no existen datos NAN, o caracteres incorrectos. Será necesario utilizar el encoder implementado en el ejercicio anterior para codificar las variables categóricas.

AllState - Configuración del entorno

En este caso la configuración utilizada es similar a la del ejercicio anterior, exceptuando algunos parámetros obvios como los de los directorios para almacenar las predicciones y los modelos generados.

La variable objetivo, que en este caso es “loss”, y por tanto hay que modificar convenientemente el parámetro TARGET.

Otra diferencia está en el parámetro TARGET_TRANSFORM en este caso indicamos que se realice la transformación logarítmica de la columna respuesta.

También hay diferencias en la métrica usada para calcular el error, en este caso se usa la métrica MAE (parámetro SCORER)

```
TARGET = "loss"
TARGET_TRANSFORM = LogTransform
SCORER = make_scorer(median_absolute_error, greater_is_better=False)
```

Con respecto a los modelos, la configuración utilizada es la siguiente, he ido descomentando en cada prueba el modelo que me interesaba y cambiando los META_PARAMETERS.

```

MODELS = {
    #('lr', LinearRegression(fit_intercept=True)),
    #('xgb', XGBRegressor(n_estimators=100, colsample_bytree=0.6, colsample_bylevel=0.6,
    #subsample=0.5, learning_rate=0.1,
    #max_depth=2, reg_alpha=0.6, min_child_weight=1)),
    ('xgb', XGBRegressor(n_estimators=1000, colsample_bytree=0.6, colsample_bylevel=0.6,
    subsample=0.5, learning_rate=0.1,
    max_depth=4, reg_alpha=0.6, min_child_weight=1)),

    # ('lasso', Lasso(fit_intercept=True)),
    # ('ridge', Ridge(fit_intercept=True)),
    # ('elastic_net', ElasticNet(fit_intercept=True)),
    # ('bayes_ridge', BayesianRidge(fit_intercept=True)),
    # ('SGD', SGDRegressor()),
    # ('random_forest', RandomForestRegressor()),
    # ('gb', GradientBoostingRegressor()),
    # ('KNN', KNeighborsRegressor(n_neighbors=10)),
}

```

AllState – Selección de modelos y entrenamiento

Los primeros resultados que obtuve fueron los siguientes:

```

Project: allstate
Date: Sun Dec 18 17:35:38 2016
Random state: 2016
Target transform: LogTransform
Parameter optimization: True
Train data shape: 188318, 133

```

Preprocess	Model	Score	STD	OP	Parameters
none	elastic_net	0.53795	0.00060	F	alpha=1.0, max_it
none	lasso	0.56923	0.00031	F	alpha=1.0, max_it
none	xgb	0.89471	0.00081	T	n_estimators=20, s
none	ridge	0.37982	0.00162	F	fit_intercept=True
none	lr	0.38345	0.00124	T	n_jobs=1, copy_X=
none	random_forest	0.49204	0.00286	T	n_estimators=1,
none	KNN	0.48267	0.00105	T	algorithm=ball_tre
none	bayes_ridge	0.37974	0.00128	F	alpha_1=1e-06, a:
none	SGD	10018498884273.96094	3215933294511.86719	F	
none	gb	0.56592	0.00012	T	loss=ls, criterion

```
Total execution time: 179.74367972215018 minutes
```


Allstate Claims Severity

Mon 10 Oct 2016 – Mon 12 Dec 2016 (24 days ago)

Your Submissions

You are submitting as part of team **inma perea**. [Make a submission »](#)

The competition deadline has already passed and you can no longer modify selections. While this competition was active, you could select up to 2 submissions. This information is provided for historical purposes only.

Submission	Files	Public Score	Private Score	Selected?
Post-Deadline: Sun, 18 Dec 2016 20:18:21 Edit description	xgb_0.894 71.csv	2087.02756	2099.55213	<input type="checkbox"/>
Post-Deadline: Wed, 04 Jan 2017 21:10:57 Edit description	bayes_ridg e_0.37974. csv	1271.09543	1279.52713	<input type="checkbox"/>

Para mejorar los resultados anteriores realicé las siguientes pruebas:

- 1) En primer lugar hice pruebas para ver si realizando operaciones entre variables, sumas, multiplicaciones mejoraba los resultados pero apenas si obtuve mejora, y algunos modelos empeoraban.
- 2) Luego probé a realizar pre-procesado, añadiendo lo siguiente por separado en el fichero config.py y tampoco obtuve mejora

```
('std', StandardScaler()),
('fs', TreeBased('extra trees regressor', 20, 190)),
```

- 3) Luego probé a aumentar el parámetro $n_estimators$ del modelo *XGBRegressor*, y la mejora fue bastante significativa. Pasando de 0.89565 a 0.35377. De modo que continúe experimentando con los parámetros de este modelo. Cambié $n_estimators$ y $learning_rate$ de forma inversa, es decir, si aumento uno disminuyo el otro. Y una vez no obtuve mejora con los anteriores comencé a aumentar el parámetro max_depth . Con esto he conseguido disminuir sustancialmente el score de 0.894 a 0.341.

AllState - Resultados

A continuación los mejores resultados de score y STD obtenidos con el framework tras el entrenamiento y la optimización de parámetros que he realizado.

```

=====
Project: allstate
Date: Fri Jan  6 13:35:07 2017
Random state: 2016
Target transform: LogTransform
Parameter optimization: True
Train data shape: 188318, 133

| Preprocess          | Model   | Score   | STD     |
| -----            | - - - - | - - - - | - - - - |
| none                | xgb     | 0.34138 | 0.00133 |

Total execution time: 11.62985545794169 minutes

```

Con los siguientes parámetros:

- missing=nan
- scale_pos_weight=1
- base_score=0.5
- n_estimators=1000
- learning_rate=0.1
- colsample_bytree=0.6
- reg_alpha=0.6
- min_child_weight=1
- subsample=0.5
- silent=True
- objective=reg:linear
- colsample_bylevel=0.6
- reg_lambda=1
- nthread=-1
- max_depth=4

Submission de Kaggle

A continuación una captura de la mejor puntuación obtenida en kaggle para este ejercicio.

Allstate Claims Severity

Mon 10 Oct 2016 – Mon 12 Dec 2016 (24 days ago)

Your Submissions

You are submitting as part of team **inma perea**.[Make a submission »](#)

The competition deadline has already passed and you can no longer modify selections. While this competition was active, you could select up to 2 submissions. This information is provided for historical purposes only.

Submission	Files	Public Score	Private Score	Selected?
Post-Deadline: Fri, 06 Jan 2017 14:17:41 Mejor score obtenido. He usado xgboost mis sing=nan scale_pos_weight=1 base_score=0.5 n_estimators=1000 learning_rate=0.1 colsample_bytree=0.6 reg_alpha=0.6 min_child_weight=1 subsample=0.5 silent=True objective=reg:linear, colsample_bylevel=0.6, reg_lambda=1 nthread=-1 max_depth=4 Edit description	xgb_0.34138.csv	1131.03929	1142.66719	<input type="checkbox"/>
Post-Deadline: Wed, 04 Jan 2017 21:18:40 Edit description	xgb_0.89471.csv	2087.02756	2099.55213	<input type="checkbox"/>

Ranking de Kaggle

A continuación una captura con el ranking privado mostrado por kaggle para las predicciones que he obtenido.

1648	↓14	Mougatine	1142.32241	22	Mon, 14 Nov 2016 17:28:02 (-31.4d)
1649	↑18	AchyutJoshi	1142.41956	4	Fri, 11 Nov 2016 11:32:37 (-5.4h)
1650	↓12	murphy	1142.44269	3	Wed, 30 Nov 2016 15:29:38 (-2d)
1651	↓6	PrashantMishra	1142.53022	7	Fri, 28 Oct 2016 19:07:25 (-2.3d)
-		inma perea	1142.66719	-	Fri, 06 Jan 2017 14:17:41 Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
1652	↑1	MikeFrantz	1142.75908	1	Mon, 28 Nov 2016 19:41:44
1653	↓31	ggautama	1142.80189	33	Tue, 29 Nov 2016 03:12:08 (-8.1d)
1654	↑9	glazed	1142.82262	1	Mon, 17 Oct 2016 17:46:57

Referencias

- [1] <https://inclass.kaggle.com/c/newcs570midtermchallenge>
- [2] <https://www.kaggle.com/c/allstateclaimsseverity>
- [3] Fichero de inspección de datos: data_inspection_kaggle.ipynb
- [4] Fichero de configuración cs570/config.py: cs570_config.py
- [5] Fichero de configuración allstate/config.py: allstate_config.py