Inteligencia de Negocio (2017-2018)

Grado en Ingeniería Informática

Universidad de Granada

Competición en Kaggle

UNIVERSIDAD DE GRANADA

Iván Rodríguez Millán

ivanrodmil@gmail.com

Índice

Índice de figuras

Índice de tablas

1. Introducción

Esta práctica ha sido llevada a cabo para la asignatura de Inteligencia de Negocio de la universidad de Granada, asignatura de cuarto curso del Grado en Ingeniería Informática. Veremos el uso de algoritmos de aprendizaje supervisado de regresión para el análisis de un dataset para predecir los precios de venta de casas haciendo uso del lenguaje de programación Python y librerías asociadas para machine learning.

La práctica consiste en competir entre los distintos alumnos de la asignatura, de tal forma que debemos con cada subida describir el objetivo de las modificaciones realizadas sobre el script y explicar por encima los parámetros empleados para los distintos algoritmos.

Para la realización de la práctica se utilizará el lenguaje de programación Python en su versión 3.6.3 [**?**].Además nos ayudaremos de librerías como Matplotlib `https://matplotlib.org/`, Sklearn `http://scikit-learn.org/stable/`, Seaborn `https://seaborn.pydata.org/` y Scipy `https://www.scipy.org/`.

Para este estudio se han utilizado los siguientes algoritmos de regresión:

- KNN.
- Random Forest.
- Gradient Boosting.
- Support Vector Regressor.
- Decision Tree.
- Elastic Net.
- Xgboost.

El conjunto de datos consta de 79 variables sobre las características de una casa, desde si la vivienda está en zona residencia, zona de agricultura, etc hasta datos como el garaje o la miscelanea.

Principalmente en esta práctica se abordará el problema haciendo un estudio lo mas escrupuloso posible, reflejando tablas con datos estadísticos la mayor cantidad de información posible.

Por último cabe decir que anexo a esta documentación se proporcionan los script realizados para cada subida, en donde para cada script se guarda la fecha y el orden seguido dentro de esa fecha para su realización.

2. Progreso de la competición

En este apartado se expondrá el progreso llevado a cabo para la competición en Kaggle, de tal forma que para hacerlo todo más ameno, se introducirán partes del código y la tabla correspondiente a cada subida con datos como el RMSE, Score obtenido ó la posición obtenida. A su vez, como material anexo a esta documentación se tienen los script en Python realizados para la práctica y los ficheros .csv que se utilizaron para la subida.

2.1. Subida1

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
23/12/2017	12:00	1038	0.12657	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies	Elastic Net	Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.119214198337
					Gradient Boosting	Estimadores con 3000 maxima profundidad con 3 minimo muestras para partir nodo con 10	0.0763571532594

Tabla 2.1: Datos asociados a cada subida 1 a la plataforma Kaggle.

En esta primera subida se hará uso del script que nos proporcionó el profesor para la iniciación en algoritmos de Regresión. A continuación se mostrará el script.

Es importante entender el script ya que en mi caso las mejoras realizadas se basan en modificaciones sobre este primer fichero de prueba. Mejoras que como veremos en algunos casos son modificaciones del preprocesamiento y en otros casos son modificaciones de los algoritmos o directamente cambiar un algoritmo por otro.

Listing 1: Primer Script

```
1  # -*- coding: utf-8 -*-
2
3  # Adding needed libraries and reading data
4  import pandas as pd
5  import numpy as np
6  import matplotlib.pyplot as plt
7  import seaborn as sns
8  from sklearn import ensemble, tree, linear_model
9  from sklearn.model_selection import train_test_split,
10 cross_val_score
11 from sklearn.metrics import r2_score, mean_squared_error
12 from sklearn.utils import shuffle
13
14 import warnings
15 warnings.filterwarnings('ignore')
16
```

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

train.head()

#Checking for missing data
NAs = pd.concat([train.isnull().sum(),
test.isnull().sum()], axis=1, keys=['Train', 'Test'])
print(NAs[NAs.sum(axis=1) > 0])

# Prints R2 and RMSE scores
def get_score(prediction, lables):
print('R2: {}'.format(r2_score(prediction, lables)))
print('RMSE: {}'.format(
np.sqrt(mean_squared_error(prediction, lables))))

# Shows scores for train and validation sets
def train_test(estimator, x_trn, x_tst, y_trn, y_tst):
prediction_train = estimator.predict(x_trn)
# Printing estimator
print(estimator)
# Printing train scores
get_score(prediction_train, y_trn)
prediction_test = estimator.predict(x_tst)
# Printing test scores
print("Test")
get_score(prediction_test, y_tst)

# Spliting to features and lables and deleting variable
I don't need
train_labels = train.pop('SalePrice')

features = pd.concat([train, test], keys=['train', 'test'])

# I decided to get rid of features that have more
than half of missing information or do not correlate
to SalePrice
features.drop(['Utilities', 'RoofMatl', 'MasVnrArea',
'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'Heating',
'LowQualFinSF',
'BsmtFullBath', 'BsmtHalfBath', 'Functional', 'GarageYrBlt',
'GarageArea', 'GarageCond', 'WoodDeckSF',
'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
'ScreenPorch', 'PoolArea', 'PoolQC', 'Fence',
'MiscFeature', 'MiscVal'],
axis=1, inplace=True)

# MSSubClass as str
features['MSSubClass'] = features['MSSubClass'].astype(str)
```

```
66
67  # MSZoning NA in pred. filling with most popular values
68  features['MSZoning'] = features['MSZoning']
69  .fillna(features['MSZoning'].mode()[0])
70
71  # LotFrontage  NA in all. I suppose NA means 0
72  features['LotFrontage'] = features['LotFrontage']
73  .fillna(features['LotFrontage'].mean())
74
75  # Alley  NA in all. NA means no access
76  features['Alley'] = features['Alley'].fillna('NOACCESS')
77
78  # Converting OverallCond to str
79  features.OverallCond = features.OverallCond.astype(str)
80
81  # MasVnrType NA in all. filling with most popular values
82  features['MasVnrType'] = features['MasVnrType']
83  .fillna(features['MasVnrType'].mode()[0])
84
85  # BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2
86  # NA in all. NA means No basement
87  for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure',
88   'BsmtFinType1', 'BsmtFinType2'):
89  features[col] = features[col].fillna('NoBSMT')
90
91  # TotalBsmtSF  NA in pred. I suppose NA means 0
92  features['TotalBsmtSF'] = features['TotalBsmtSF'].fillna(0)
93
94  # Electrical NA in pred. filling with most popular values
95  features['Electrical'] = features['Electrical']
96  .fillna(features['Electrical'].mode()[0])
97
98  # KitchenAbvGr to categorical
99  features['KitchenAbvGr'] = features['KitchenAbvGr'].astype(str)
100
101 # KitchenQual NA in pred. filling with most popular values
102 features['KitchenQual'] = features['KitchenQual']
103 .fillna(features['KitchenQual'].mode()[0])
104
105 # FireplaceQu  NA in all. NA means No Fireplace
106 features['FireplaceQu'] = features['FireplaceQu']
107 .fillna('NoFP')
108
109 # GarageType, GarageFinish, GarageQual
110 NA in all. NA means No Garage
111 for col in ('GarageType', 'GarageFinish', 'GarageQual'):
112 features[col] = features[col].fillna('NoGRG')
113
114 # GarageCars  NA in pred. I suppose NA means 0
```

```
features['GarageCars'] = features['GarageCars'].fillna(0.0)

# SaleType NA in pred. filling with most popular values
features['SaleType'] = features['SaleType']
.fillna(features['SaleType'].mode()[0])

# Year and Month to categorical
features['YrSold'] = features['YrSold'].astype(str)
features['MoSold'] = features['MoSold'].astype(str)

# Adding total sqfootage feature and removing
Basement, 1st and 2nd floor features
features['TotalSF'] = features['TotalBsmtSF'] +
features['1stFlrSF'] + features['2ndFlrSF']
features.drop(['TotalBsmtSF', '1stFlrSF', '2ndFlrSF'],
axis=1, inplace=True)

# Our SalesPrice is skewed right (check plot below).
I'm logtransforming it.
plt.figure(1)
plt.clf()
ax = sns.distplot(train_labels)

## Log transformation of labels
train_labels = np.log(train_labels)

## Now it looks much better
plt.figure(2)
plt.clf()
ax = sns.distplot(train_labels)

## Standardizing numeric features
numeric_features = features.loc[:,['LotFrontage', 'LotArea',
'GrLivArea', 'TotalSF']]
numeric_features_standardized =
(numeric_features - numeric_features.mean())
/numeric_features.std()

#ax = sns.pairplot(numeric_features_standardized)

# Getting Dummies from Condition1 and Condition2
conditions = set([x for x in features['Condition1']] +
[x for x in features['Condition2']])
dummies = pd.DataFrame(data=
np.zeros((len(features.index), len(conditions))),
index=features.index, columns=conditions)
for i, cond in enumerate(zip(features['Condition1'],
features['Condition2'])):
dummies.ix[i, cond] = 1
```

```
features = pd.concat([features, dummies.add_prefix
('Condition_')], axis=1)
features.drop(['Condition1', 'Condition2'], axis=1,
inplace=True)

# Getting Dummies from Exterior1st and Exterior2nd
exteriors = set([x for x in features['Exterior1st']] +
[x for x in features['Exterior2nd']])
dummies = pd.DataFrame(data=np.zeros((len(features.index),
len(exteriors))),
index=features.index, columns=exteriors)
for i, ext in enumerate(zip(features['Exterior1st'],
features['Exterior2nd'])):
dummies.ix[i, ext] = 1
features = pd.concat([features,
dummies.add_prefix('Exterior_')], axis=1)
features.drop(['Exterior1st', 'Exterior2nd',
'Exterior_nan'], axis=1, inplace=True)

# Getting Dummies from all other categorical vars
for col in features.dtypes[features.dtypes == 'object'].index:
for_dummy = features.pop(col)
features = pd.concat([features,
pd.get_dummies(for_dummy, prefix=col)], axis=1)

### Copying features
features_standardized = features.copy()

### Replacing numeric features by standardized values
features_standardized.update(numeric_features_standardized)

### Splitting features
train_features = features.loc['train'].drop('Id', axis=1)
.select_dtypes(include=[np.number]).values
test_features = features.loc['test'].drop('Id', axis=1)
.select_dtypes(include=[np.number]).values

### Splitting standardized features
train_features_st = features_standardized.loc['train'].drop('Id',
 axis=1).select_dtypes(include=[np.number]).values

test_features_st = features_standardized.loc['test'].drop('Id',
 axis=1).select_dtypes(include=[np.number]).values

### Shuffling train sets
train_features_st, train_features, train_labels =
shuffle(train_features_st, train_features, train_labels,
random_state = 5)

```

```
### Splitting
x_train, x_test, y_train, y_test =
train_test_split(train_features, train_labels,
test_size=0.1, random_state=200)
x_train_st, x_test_st, y_train_st, y_test_st =
train_test_split(train_features_st, train_labels,
test_size=0.1, random_state=200)

'''
Elastic Net
'''
ENSTest = linear_model.ElasticNetCV(alphas=
[0.0001, 0.0005, 0.001, 0.01, 0.1, 1, 10], l1_ratio=
[.01, .1, .5, .9, .99], max_iter=5000).
fit(x_train_st, y_train_st)
train_test(ENSTest, x_train_st,
x_test_st, y_train_st, y_test_st)

# Average R2 score and standard deviation
of 5-fold cross-validation
scores = cross_val_score(ENSTest, train_features_st,
train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" %
(scores.mean(), scores.std() * 2))

'''
Gradient Boosting
'''

GBest = ensemble.GradientBoostingRegresso
r(n_estimators=3000, learning_rate=0.05, max_depth=3,
max_features='sqrt',
min_samples_leaf=15, min_samples_split=10, loss='huber').
fit(x_train, y_train)
train_test(GBest, x_train, x_test,
y_train, y_test)

# Average R2 score and standart deviation
of 5-fold cross-validation
scores = cross_val_score(GBest,
train_features_st, train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" %
(scores.mean(), scores.std() * 2))

# Retraining models
GB_model = GBest.fit(train_features, train_labels)
ENST_model = ENSTest.fit(train_features_st, train_labels)

## Getting our SalePrice estimation
```

```
Final_labels = (np.exp(GB_model.predict(test_features)) +
np.exp(ENST_model.predict(test_features_st))) / 2

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels}).
to_csv('submission.csv', index =False)
```

2.1.1. Preprocesamiento y Algoritmos usados

En este primer script vemos que en un principio tenemos dos dataset, Train y Test. En pasos posteriores se concatenan en un dataframe llamado features, donde previamente se eliminan una serie de variables, estas variables eliminadas juegan un papel fundamental en posteriores subidas, ya que en mi caso intentaré imputar valores perdidos para intentar mejorar algo las métricas.

Más adelante se puede apreciar como se imputa valores perdidos a algunas variables, otras se convierten a variables binarias y otras a variables categóricas.

Al final consiste en tener los datos lo más limpios posibles para ofrecerlos al regresor correspondiente y obtener datos de calidad. En esta primera subida se han usado los algoritmos Elastic Net y Gradient Boosting.
Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.

- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.2. Subida2

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
25/12/2017	13:00	1038	0.14136	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies	Elastic Net	Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.119214198337
					Random Forest	Estimadores con 4000 maxima profundidad con 4 minimo muestras para partir nodo con 10	0.13421400424

Tabla 2.2: Datos asociados a cada subida 2 a la plataforma Kaggle.

Modificaciones realizadas para la segunda subida:

Listing 2: Segunda modificación

```
'''
Random Forest
'''

RFest = ensemble.RandomForestRegressor
(n_estimators=4000, max_depth=4, max_features='sqrt',
min_samples_leaf=15, min_samples_split=10)
.fit(x_train, y_train)

train_test(RFest, x_train, x_test, y_train, y_test)

# Average R2 score and standart deviation
of 5-fold cross-validation
scores = cross_val_score(RFest, train_features_st,
 train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" %
(scores.mean(), scores.std() * 2))

# Retraining models
RF_model = RFest.fit(train_features, train_labels)
ENST_model = ENSTest.fit(train_features_st, train_labels)

## Getting our SalePrice estimation
Final_labels = (np.exp(RF_model.predict(test_features))
+ np.exp(ENST_model.predict(test_features_st))) / 2

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels})
.to_csv('submissionRandomForest.csv', index =False)
```

2.2.1. Preprocesamiento y Algoritmos usados

En este segundo script se lleva a cabo el mismo procesamiento visto para la primera subida 2.1, la modificación que se ha realizado es la de cambiar el algoritmo Gradient

Boosting por el Random Forest para ver si se produce alguna mejora por muy ínfima que sea. Sin embargo vemos como no se produce ninguna mejora y empeora los resultados. Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Random Forest:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Random Forest).

2.3. Subida3

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
25/12/2017	16:00	1038	0.14448	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.11921419305625844
					Decision Tree	Decision Tree: Máxima profundidad de 10 Mínimo número de muestras para la partición de un nodo es de 15	0.18336602233360735

Tabla 2.3: Datos asociados a cada subida 3 a la plataforma Kaggle.

Modificaciones realizadas para la tercera subida:

Listing 3: Tercera modificación

```
'''
Decision Tree
'''

DTest = tree.DecisionTreeRegressor(max_depth=100
, random_state=0, min_samples_leaf=10,
min_samples_split=15, max_features='sqrt')
.fit(x_train, y_train)

train_test(DTest, x_train, x_test, y_train, y_test)

# Average R2 score and standart deviation
of 5-fold cross-validation
scores = cross_val_score(DTest, train_features_st,
train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" %
(scores.mean(), scores.std() * 2))

# Retraining models
DT_model = DTest.fit(train_features, train_labels)
ENST_model = ENSTest.fit(train_features_st,
train_labels)

## Getting our SalePrice estimation
Final_labels = (np.exp(DT_model.predict(test_features))
+ np.exp(ENST_model.predict(test_features_st))) / 2

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels})
.to_csv('submissionDecisionTree.csv', index =False)
```

2.3.1. Preprocesamiento y Algoritmos usados

En este segundo script se lleva a cabo el mismo procesamiento visto para la primera subida 2.1, la modificación que se ha realizado es la de cambiar el algoritmo Random Forest por el Decision Tree para ver si se produce alguna mejora por muy ínfima que sea. Sin embargo vemos como no se produce ninguna mejora y empeora los resultados.
Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Decision Tree:

- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 10.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 15.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Decision Tree).

2.4. Subida4

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
25/12/2017	16:30	1038	0.24298	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.11921418466703167
					Support Vector Regressor	Support Vector Regressor : Número máximo de iteraciones de 3000 Epsilon igual a 0.2 rbf como kernel	0.17413587923793689

Tabla 2.4: Datos asociados a la subida 4 a la plataforma Kaggle.

Modificaciones realizadas para la cuarta subida:

Listing 4: Cuarta modificación

```
'''
SVR
'''

SVRest = SVR(C=1.0, cache_size=200, coef0=0.0,
degree=3, epsilon=0.2, gamma='auto',
kernel='rbf', max_iter=3000, shrinking=True, tol=0.001,
verbose=False).fit(x_train, y_train)

train_test(SVRest, x_train, x_test, y_train, y_test)

# Average R2 score and standart deviation
of 5-fold cross-validation
scores = cross_val_score(SVRest, train_features_st,
 train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" %
(scores.mean(), scores.std() * 2))

# Retraining models
SVR_model = SVRest.fit(train_features, train_labels)
ENST_model = ENSTest.fit(train_features_st, train_labels)

## Getting our SalePrice estimation
Final_labels = (np.exp(SVR_model.predict(test_features)) +
 np.exp(ENST_model.predict(test_features_st))) / 2

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels})
.to_csv('submissionSVR.csv', index =False)
```

2.4.1. Preprocesamiento y Algoritmos usados

En este segundo script se lleva a cabo el mismo procesamiento visto para la primera subida 2.1, la modificación que se ha realizado es la de cambiar el algoritmo Decision

Tree por el Support Vector Regressor para ver si se produce alguna mejora por muy ínfima que sea. Sin embargo vemos como no se produce ninguna mejora y empeora los resultados. En este caso podemos ver un gran aumento con respecto a versiones anteriores en el score, indicador claro de que este no es el camino que debemos llevar para afianzarnos en posiciones mejores.

Claramente vemos como el algoritmo Support Vector Regressor tiene un RMSE alto con respecto a otros algoritmos utilizados anteriormente, solamente superado por el Árbol de decisión. Esto y el score final de esta subida fueron claros indicativos de que debíamos cambiar la estrategia de ir observando distintos comportamientos de los distintos algoritmos. Lo que necesitaba era afianzar un algoritmo modificando el preprocesado previo. Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Support Vector Regressor:

- Maxiter: Número máximo de iteraciones. En este caso serán unas 3000.
- Kernel: Especifica el tipo de kernel para el modelo. En este caso será rbf.
- CacheSize: Especifica el tamaño de la cache del kernel. En nuestro caso será de 200.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Support Vector Regressor).

2.5. Subida5

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
26/12/2017	10:00	935	0.12520	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.119214198337
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.0568918247635

Tabla 2.5: Datos asociados a la subida 5 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 5: Quinta modificación

```
'''
Gradient Boosting
'''

GBest = ensemble.GradientBoostingRegressor
(n_estimators=3500, learning_rate=0.05, max_depth=5,
max_features='sqrt',
min_samples_leaf=15, min_samples_split=10,
loss='huber').fit(x_train, y_train)
```

2.5.1. Preprocesamiento y Algoritmos usados

En este quinto script intentamos poner la primera piedra de lo que será una serie de subidas con el algoritmo Gradient Boosting, sin modificar el preprocesado previo, probamos a mejorar el algoritmo con respecto a la primera subida, en este caso como se puede apreciar en la tabla 2.5 se produce una disminución sustancial de la métrica RMSE con respecto a la tabla 2.1. Esto es un indicativo de que con los datos actuales sin preprocesar se comporta mejor la nueva configuración del algoritmo Gradient Boosting.

Toda esta mejora viene a raíz de aumentar el número de estimadores y la profundidad máxima para el estimador individual como se puede apreciar en el código anterior.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3500. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 5.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.6. Subida6

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
26/12/2017	12:00	935	0.12664	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies salvo para las características Exterior, en tal caso se hace imputación de valores	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.11924691844805038
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.05525372426994724

Tabla 2.6: Datos asociados a la subida 6 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 6: Sexta modificación

```
'''for i, ext in enumerate(zip(features['Exterior1st'],
features['Exterior2nd'])):
dummies.ix[i, ext] = 1
features = pd.concat([features,
dummies.add_prefix('Exterior_')], axis=1)
features.drop(['Exterior1st', 'Exterior2nd',
'Exterior_nan'], axis=1, inplace=True)'''

features['Exterior1st'] = features['Exterior1st']
.fillna(features['Exterior1st'].mode()[0])
features['Exterior2nd'] = features['Exterior2nd']
.fillna(features['Exterior2nd'].mode()[0])
```

2.6.1. Preprocesamiento y Algoritmos usados

Para esta sexta subida se empieza a tocar parte del preprocesamiento, en este caso pasamos de binarizar las características Exterior por imputar valores en los casos donde residan valores NA. Nos dió peores resultados que para la anterior subida como se puede apreciar en el score final, por tanto no se prosiguió por este camino.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3500. Un número grande generalmente para el Gradient Boosting da buen resultado.

- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 5.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.7. Subida7

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
26/12/2017	12:30	704	0.12120	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.10919518132276353
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.05330189080647167

Tabla 2.7: Datos asociados a la subida 7 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 7: Sexta modificación

```
1
2  features.drop(['Alley', 'PoolQC', 'Fence', 'FireplaceQu',
3  'MiscVal', 'MiscFeature', 'LotFrontage','GarageYrBlt',
4  'GarageArea', 'GarageCond',
5  'MasVnrArea','MasVnrType', 'GarageQual'],
6  axis=1, inplace=True)
7
8  features['Utilities'] = features['Utilities'].fillna('AllPub')
9
10 features['RoofMatl'] = features['RoofMatl'].fillna('CompShg')
11
12 features['BsmtFinSF1'] = features['BsmtFinSF1'].fillna(0)
13
14 features['BsmtFinSF2'] = features['BsmtFinSF2'].fillna(0)
15
16 features['BsmtUnfSF'] = features['BsmtUnfSF'].fillna(0)
17
18 features['Heating'] = features['Heating'].fillna('GasA')
19
20 features['LowQualFinSF'] = features['LowQualFinSF'].fillna(0)
21
22 features['BsmtFullBath'] = features['BsmtFullBath'].fillna(0)
23
24 features['BsmtHalfBath'] = features['BsmtHalfBath'].fillna(0)
25
26 features['Functional'] = features['Functional'].fillna('Typ')
27
28 features['WoodDeckSF'] = features['WoodDeckSF'].fillna(0)
29
30 features['OpenPorchSF'] = features['OpenPorchSF'].fillna(0)
31
32 features['EnclosedPorch'] = features['EnclosedPorch'].fillna(0)
33
34 features['3SsnPorch'] = features['3SsnPorch'].fillna(0)
35
36 features['ScreenPorch'] = features['ScreenPorch'].fillna(0)
```

```
37
38 features['PoolArea'] = features['PoolArea'].fillna(0)
39
40 features['PoolArea'] = features['PoolArea'].fillna(0)
41
42 # MSSubClass as str
43 features['MSSubClass'] = features['MSSubClass'].astype(str)
44
45 # MSZoning NA in pred. filling with most popular values
46 features['MSZoning'] = features['MSZoning']
47 .fillna(features['MSZoning'].mode()[0])
48
49 # LotFrontage  NA in all. I suppose NA means 0
50 #features['LotFrontage'] = features['LotFrontage']
51 .fillna(features['LotFrontage'].mean())
52
53 # Alley  NA in all. NA means no access
54 #features['Alley'] = features['Alley'].fillna('NOACCESS')
55
56 # Converting OverallCond to str
57 features.OverallCond = features.OverallCond.astype(str)
58
59 # MasVnrType NA in all. filling with most popular values
60 #features['MasVnrType'] = features['MasVnrType']
61 .fillna(features['MasVnrType'].mode()[0])
62
63 # BsmtQual, BsmtCond, BsmtExposure,
64 BsmtFinType1, BsmtFinType2
65 # NA in all. NA means No basement
66 for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure',
67 'BsmtFinType1', 'BsmtFinType2'):
68 features[col] = features[col].fillna('NoBSMT')
69
70 # TotalBsmtSF  NA in pred. I suppose NA means 0
71 features['TotalBsmtSF'] =
72 features['TotalBsmtSF'].fillna(0)
73
74 # Electrical NA in pred. filling with most popular values
75 features['Electrical'] = features['Electrical']
76 .fillna(features['Electrical'].mode()[0])
77
78 # KitchenAbvGr to categorical
79 features['KitchenAbvGr'] = features['KitchenAbvGr'].astype(str)
80
81 # KitchenQual NA in pred. filling with most popular values
82 features['KitchenQual'] = features['KitchenQual']
83 .fillna(features['KitchenQual'].mode()[0])
84
85 # FireplaceQu  NA in all. NA means No Fireplace
```

```
#features['FireplaceQu'] = features['FireplaceQu'
].fillna('NoFP')

# GarageType, GarageFinish, GarageQual
NA in all. NA means No Garage
#for col in ('GarageType', 'GarageFinish',
'GarageQual'):
#    features[col] = features[col]
.fillna('NoGRG')

# GarageCars  NA in pred. I suppose NA means 0
features['GarageCars'] = features['GarageCars']
.fillna(0.0)

# SaleType NA in pred. filling with most popular values
features['SaleType'] = features['SaleType']
.fillna(features['SaleType'].mode()[0])

# Year and Month to categorical
features['YrSold'] = features['YrSold'].astype(str)
features['MoSold'] = features['MoSold'].astype(str)

# Adding total sqfootage feature and removing
Basement, 1st and 2nd floor features
features['TotalSF'] = features['TotalBsmtSF'] +
features['1stFlrSF'] + features['2ndFlrSF']
features.drop(['TotalBsmtSF', '1stFlrSF', '2ndFlrSF'],
axis=1, inplace=True)
```

2.7.1. Preprocesamiento y Algoritmos usados

En esta subida llevamos un poco más al extremo la imputación de valores, y es que básicamente lo que hacemos es eliminar menos características (features) para así logar un avance imputando valores, ya sea introduciendo valores más comunes para la característica en concreto o introduciendo medias o valores máximos. Encima de esta subsección tenemos el trozo de código que se introduce.

Gracias a esto se puede ver un gran avance, y es que como todo hacía apuntar, el mayor escollo estará en el preprocesamiento. Con este paso bajamos hasta la posición 704 con un grandísimo score en comparación con subidas anteriores.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización.

En nuestro caso se puede ver que se le pasa una lista con varias alphas.

- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3500. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 5.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.8. Subida8

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
26/12/2017	13:00	704	0.12178	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.10917301872807618
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.051617037891679915

Tabla 2.8: Datos asociados a la subida 8 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 8: Octava modificación

```
numeric_features = features.loc[:,['LotArea', 'TotalSF']]
numeric_features_standardized =
(numeric_features - numeric_features.mean())
/numeric_features.std()
```

2.8.1. Preprocesamiento y Algoritmos usados

En esta subida no se estandarizan las características GrLivArea y LotFrontage, en este segundo caso eliminamos la característica como se puede ver en el script de la subida anterior. No terminamos de mejorar la anterior subida, pero nos quedamos cerca, por lo tanto son resultados esperanzadores para seguir por este camino.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3500. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 5.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.

- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.9. Subida9

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
26/12/2017	16:00	688	0.12099	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.1091951814069412
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07375388317842049

Tabla 2.9: Datos asociados a la subida 9 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 9: Novena modificación

```
'''
Gradient Boosting
'''

GBest = ensemble.GradientBoostingRegressor
(n_estimators=3000, learning_rate=0.05, max_depth=3,
max_features='sqrt', min_samples_leaf=15,
 min_samples_split=10, loss='huber')
.fit(x_train, y_train)
train_test(GBest, x_train, x_test, y_train, y_test)

# Average R2 score and standart deviation of
5-fold cross-validation
scores = cross_val_score(GBest, train_features_st,
train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)"
% (scores.mean(), scores.std() * 2))

# Retraining models
GB_model = GBest.fit(train_features, train_labels)
ENST_model = ENSTest.fit(train_features_st, train_labels)

## Getting our SalePrice estimation
Final_labels = (np.exp(GB_model.predict(test_features))
+ np.exp(ENST_model.predict(test_features_st))) / 2

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels})
.to_csv('submissionGradientBoosted.csv', index =False)

#GarageCars  NA in pred. I suppose NA means 0
features['GarageCars'] = features['GarageCars'].fillna(0.0)

## Standardizing numeric features
numeric_features = features.loc[:,['LotArea', 'GrLivArea',
'TotalSF']]
```

```
numeric_features_standardized =
(numeric_features - numeric_features.mean())
/numeric_features.std()
```

2.9.1. Preprocesamiento y Algoritmos usados

En esta subida se realizan dos cosas, en primer lugar modificamos los parámetros del algoritmo Gradient Boosting en busca de mejores resultados, y en segundo lugar se vuelve a realizar el estandarizado numérico de la característica GrLivArea.

Estos cambios dan muy buenos resultados, y es que básicamente estamos volviendo a configurar el algoritmo Gradient Boosting como estaba al principio, ya que tras mucho leer alguno de los kernels mostrados por el profesor se nos aseguraba que esta configuración era la idónea para este ejercicio empírico de analásis con algoritmos de regresión.

Cabe decir que a la variable GarageCars se le imputan los valores correspondiente por NA. En este caso el valor de 0(Que mostraba que no tenía garage para coche).

Con esta modificación conseguimos establecernos en la posición 688 bajando claramente el score de 0.12100.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.

- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.10. Subida10

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
26/12/2017	17:00	688	0.12177	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.10897258851186785
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07209852218733985

Tabla 2.10: Datos asociados a la subida 10 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 10: Décima modificación

```
# GarageType, GarageFinish, GarageQual  NA in all. NA means No Garage
#for col in ('GarageType', 'GarageFinish', 'GarageQual'):
#    features[col] = features[col].fillna('NoGRG')

# GarageCars  NA in pred. I suppose NA means 0
#features['GarageCars'] = features['GarageCars'].fillna(0.0)
for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
features[col] = features[col].fillna(0)
```

2.10.1. Preprocesamiento y Algoritmos usados

Para esta subida se toca el preprocesamiento, y en particular pasamos de rellenar con ceros las características GarageYrBlt, GarageArea y GarageCars. Las características GarageType, GarageFinish y GarageQual se eliminan.

No se alcanzan tan buenos resultados como pensábamos pero tampoco se hace un retroceso en la investigación.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.

- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.11. Subida11

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
27/12/2017	19:00	601	0.12051	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.10871744415804033
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07106525465111262

Tabla 2.11: Datos asociados a la subida 11 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 11: Undécima modificación

```
features.drop(['Alley', 'PoolQC', 'Fence', 'FireplaceQu',
 'MiscVal', 'MiscFeature', 'LotFrontage',
'MasVnrArea', 'MasVnrType'],
axis=1, inplace=True)

features['Utilities'] = features['Utilities'].fillna('AllPub')

for col in ('GarageYrBlt', 'GarageArea', 'GarageCars',
 'GarageCond', 'GarageQual'):
features[col] = features[col].fillna(0)
```

2.11.1. Preprocesamiento y Algoritmos usados

Para la undécima subida cambiamos las características a eliminar, decidimos que los qué tienen pocos valores perdidos como es el caso de Utilities, GarageQual y GarageCond no los vamos a eliminar y si a meter valores donde los tengan a NA como se puede apreciar en la parte de script introducida en la documentación.

En este caso avanzamos hasta la posición 601 con un score bastante bueno con respecto a la subida 2.9. Y es que vamos viendo, como el tratar los valores perdidos o en este caso valores como NA en vez de eliminar las características que los poseen, nos da una cierta ventaja.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.12. Subida12

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
27/12/2017	20:00	601	0.12205	Eliminación de 2 características, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.1085382484492128
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07153527836510816

Tabla 2.12: Datos asociados a la subida 12 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 12: Duodécima modificación

```
features.drop(['MiscVal'], axis=1, inplace=True)

features["MasVnrType"] = features["MasVnrType"].fillna("None")
features["MasVnrArea"] = features["MasVnrArea"].fillna(0)

features["PoolQC"] = features["PoolQC"].fillna("None")

features["MiscFeature"] = features["MiscFeature"].fillna("None")

features["Alley"] = features["Alley"].fillna("None")

features["Fence"] = features["Fence"].fillna("None")

features["FireplaceQu"] = features["FireplaceQu"].fillna("None")

features["LotFrontage"] = features.groupby("Neighborhood")["LotFrontage"].trans
lambda x: x.fillna(x.median()))

features = features.drop(['Utilities'], axis=1)
```

2.12.1. Preprocesamiento y Algoritmos usados

Para la subida duodécima se eliminan tan solo las características Utilities y MiscVal. En las restantes características que pasamos de no eliminarlas a imputar valores perdidos básicamente se introduce la cadena None y en el caso de la característica LotFrontage la media.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.

- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.13. Subida13

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
29/12/2017	11:00	613	0.12382	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.10853824692912997
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07085867327998868

Tabla 2.13: Datos asociados a la subida 13 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 13: Decimotercera modificación

```
1 features.drop(features[features['Id'] == 1299].index)
2 features.drop(features[features['Id'] == 524].index)
```

2.13.1. Preprocesamiento y Algoritmos usados

Para la subida decimotercera se eliminan dos instancias identificadas por los ID 1299 y 524. Estas instancias se corresponde con valores con outliers dentro de la característica denominada GrLivArea. Esta información ha sido obtenida a partir de la siguiente imagen, en donde se pueden ver dos instancias alejadas con respecto a las demas en la parte baja a la derecha:

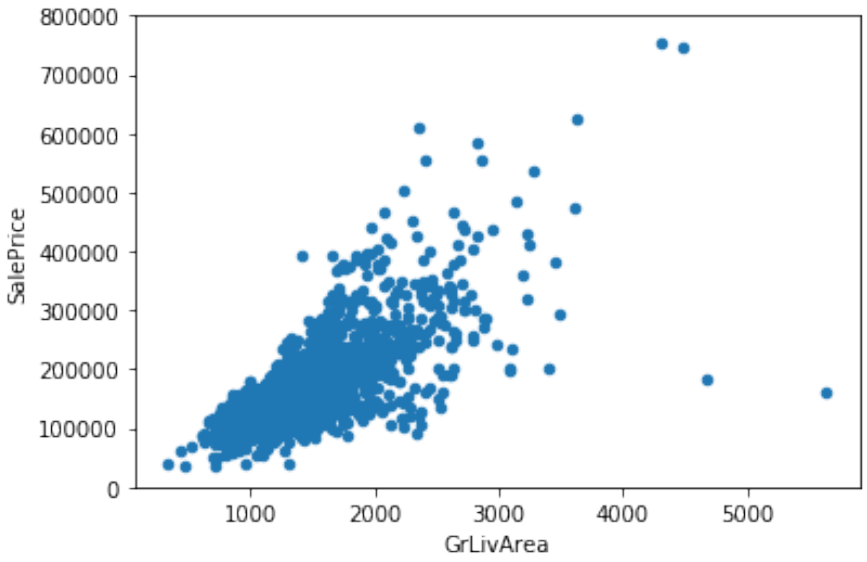

Figura 2.1: Imagen comparativa del precio de venta con la característica GrLivArea.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.14. Subida14

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
01/01/2018	11:00	630	0.16925	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.10853824692912997
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07085867327998868
					Random Forest	Random Forest: Estimadores con 4000 maxima profundidad con 4 y minimo muestras para partir nodo con 10.	0.19136166739402494

Tabla 2.14: Datos asociados a la subida 14 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 14: Duodécima modificación

```
'''
Random Forest
'''

RFest = ensemble.RandomForestRegressor(n_estimators=4000
, max_depth=4, max_features='sqrt',
min_samples_leaf=15, min_samples_split=10).fit(x_train, y_train)

train_test(RFest, x_train, x_test, y_train, y_test)

# Average R2 score and standart deviation of
5-fold cross-validation
scores = cross_val_score(RFest, train_features_st,
train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)"
% (scores.mean(), scores.std() * 2))

# Retraining models
RF_model = RFest.fit(train_features, train_labels)
ENST_model = ENSTest.fit(train_features_st, train_labels)

## Getting our SalePrice estimation
Final_labels = (np.exp(RF_model.predict(test_features))
+ np.exp(ENST_model.predict(test_features_st)) +
np.exp(RF_model.predict(test_features))) / 3

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels})
.to_csv('submissionRandomForest.csv', index =False)
```

2.14.1. Preprocesamiento y Algoritmos usados

Para la decimocuarta subida básicamente lo que hacemos es dar entrada a un nuevo algoritmo a nuestra ecuación. Como era de preveer el algoritmo Random Forest no da buenos resultados junto con los algoritmos Elastic Net y Gradient Boosting.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Como parámetros más llamativos para esos algoritmos tenemos, para Random Forest:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 4000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 4.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.

- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los tres algoritmos de regresión utilizados (Elastic Net, Gradient Boosting y Random forest).

2.15. Subida15

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
02/01/2018	17:00	627	0.12415	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.09833172173468259

Tabla 2.15: Datos asociados a la subida 15 a la plataforma Kaggle.

Modificaciones realizadas para la decimoquinta subida:

Listing 15: Decimoquinta modificación

```
train = train [-((train.GrLivArea > 4000) &
(train.SalePrice < 300000))]

'''
Elastic Net
'''

ENSTest = linear_model.ElasticNetCV(alphas=
[0.0001, 0.0005, 0.001, 0.01, 0.1, 1, 10],
 l1_ratio=[.01, .1, .5, .9, .99],
max_iter=5000).fit(x_train_st, y_train_st)
train_test(ENSTest, x_train_st, x_test_st, y_train_st,
y_test_st)

# Average R2 score and standard deviation of
5-fold cross-validation
scores = cross_val_score(ENSTest, train_features_st,
train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
scores.std() * 2))
ENST_model = ENSTest.fit(train_features_st, train_labels)

## Getting our SalePrice estimation
Final_labels = np.exp(ENST_model.predict(test_features_st))

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels}).
to_csv('submissionElasticNet.csv', index=False)
```

2.15.1. Preprocesamiento y Algoritmos usados

Para la decimoquinta subida lo que hacemos es dar entrada a un único algoritmo, el Elastic Net. El resultado final no es malo, simplemente no mejora lo que ya teníamos. Aparte de esto se realiza una eliminación de Outliers más precisa

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Al final del todo en el script se puede ver como la estimación final es la estimación del algoritmo de regresión Elastic Net.

2.16. Subida16

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
02/01/2018	23:00	628	0.12095	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.09833172178187898
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.060417418639638586

Tabla 2.16: Datos asociados a la subida 16 a la plataforma Kaggle.

Modificaciones realizadas para la decimosexta subida:

Listing 16: Decimosexta modificación

```
'''
Elastic Net
'''

ENSTest = linear_model.ElasticNetCV(
alphas=[0.0001, 0.0005, 0.001, 0.01, 0.1, 1, 10],
 l1_ratio=[.01, .1, .5, .9, .99],
max_iter=5000).fit(x_train_st, y_train_st)
train_test(ENSTest, x_train_st, x_test_st,
y_train_st, y_test_st)

# Average R2 score and standard
deviation of 5-fold cross-validation
scores = cross_val_score(ENSTest,
 train_features_st, train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)"
% (scores.mean(), scores.std() * 2))

'''
Gradient Boosting
'''

GBest = ensemble.GradientBoostingRegressor
(n_estimators=3000, learning_rate=0.05, max_depth=3,
max_features='sqrt',
min_samples_leaf=15, min_samples_split=10,
loss='huber').fit(x_train,
y_train)
train_test(GBest, x_train, x_test, y_train, y_test)

# Average R2 score and standart deviation of
5-fold cross-validation
scores = cross_val_score(GBest, train_features_st,
train_labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)"
% (scores.mean(), scores.std() * 2))
```

```
37
38 # Retraining models
39 GB_model = GBest.fit(train_features, train_labels)
40 ENST_model = ENSTest.fit(train_features_st,
41 train_labels)
42
43 ## Getting our SalePrice estimation
44 # Final_labels = (np.exp(GB_model.predict(test_features))
45 + np.exp(ENST_model.predict(test_features_st))) / 2
46
47 ## Getting our SalePrice estimation
48 Final_labels = (np.exp(GB_model.predict(test_features))
49 + np.exp(ENST_model.predict(test_features_st))) / 2
50
51 ## Saving to CSV
52 pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels})
53 .to_csv('submissionGradientBoosted.csv', index=False)
```

2.16.1. Preprocesamiento y Algoritmos usados

En esta decimosexta y penúltima subida a Kaggle muestro un poco de desesperación porque noto como el algoritmo Gradient Boosting junto con el Elastic Net no terminan de mejorar con el preprocesado que se viene haciendo durante toda la práctica. Por ello como veremos en la siguiente subida lo que haremos será probar con un nuevo algoritmo hasta ahora no usado por mi.

En este caso no existe mejora alguna sobre la mejor solución subida anteriormente, por lo tanto no se mejora.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.

- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los dos algoritmos de regresión utilizados (Elastic Net y Gradient Boosting).

2.17. Subida17

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
03/01/2018	23:50	620	0.12010	Eliminación de outliers Eliminación de outliers previa en train Cambio de variables de tipo cadena a variables de tipo numérico Imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000	0.09658446906737084
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.06073413154465321
					XGBoost	Random Forest: Estimadores con 4000 maxima profundidad con 4 y minimo muestras para partir nodo con 10.	0.04349111990813433

Tabla 2.17: Datos asociados a la subida 17 a la plataforma Kaggle.

Modificaciones realizadas para la quinta subida:

Listing 17: DecimoSéptima modificación

```
train.drop(train[train["GrLivArea"] > 4000]
.index, inplace=True)

test.loc[666, "GarageQual"] = "TA"
test.loc[666, "GarageCond"] = "TA"
test.loc[666, "GarageFinish"] = "Unf"
test.loc[666, "GarageYrBlt"] = "1980"

qual_dict = {None: 0, "Po": 1, "Fa": 2,
"TA": 3, "Gd": 4, "Ex": 5}
features["ExterQual"] = features["ExterQual"].
map(qual_dict).astype(int)
features["ExterCond"] = features["ExterCond"].
map(qual_dict).astype(int)
#features["BsmtQual"] = features["BsmtQual"].
map(qual_dict).astype(int)
#features["BsmtCond"] = features["BsmtCond"].
map(qual_dict).astype(int)
features["HeatingQC"] = features["HeatingQC"].
map(qual_dict).astype(int)
features["KitchenQual"] = features["KitchenQual"].
map(qual_dict).astype(int)
#features["FireplaceQu"] = features["FireplaceQu"].
map(qual_dict).astype(int)
#features["GarageQual"] = features["GarageQual"].
map(qual_dict).astype(int)
#features["GarageCond"] = features["GarageCond"].
map(qual_dict).astype(int)


features["Functional"] = features["Functional"].map(
{None: 0, "Sal": 1, "Sev": 2, "Maj2": 3, "Maj1": 4,
"Mod": 5, "Min2": 6, "Min1": 7, "Typ": 8}).astype(int)

```

```
regrXGB = xgb.XGBRegressor(colsample_bytree=0.2,
gamma=0.0,
learning_rate=0.05,
max_depth=6,
min_child_weight=1.5,
n_estimators=7200,
reg_alpha=0.9,
reg_lambda=0.6,
subsample=0.2,
seed=42,
silent=1)

regrXGB.fit(x_train_st, y_train_st)
train_test(regrXGB, x_train_st, x_test_st,
y_train_st, y_test_st)

# Average R2 score and standard deviation of
5-fold cross-validation
scores = cross_val_score(regrXGB, train_features_st,
train_labels, cv=5)

# Retraining models
XG_model = regrXGB.fit(train_features, train_labels)
ENST_model = ENSTest.fit(train_features_st, train_labels)

## Getting our SalePrice estimation
Final_labels = (np.exp(GB_model.predict(test_features))
+ np.exp(ENST_model.predict(test_features_st))
+ np.exp(XG_model.predict(test_features))) / 3

## Saving to CSV
pd.DataFrame({'Id': test.Id, 'SalePrice': Final_labels})
.to_csv('submissionXGBoost.csv', index=False)
```

2.17.1. Preprocesamiento y Algoritmos usados

Para la última subida en el último segundo a Kaggle lo que hacemos es la eliminación de outliers en el dataset de train, después cambiamos algunas características de tipo cadena a tipo numérico, y posteriormente se aplica junto con Elastic Net y Gradient Boosting, el algoritmo XGBoost.

De esta forma al predecir la variable precio, nosotros la indicamos como el promedio entre las 3 predicciones de los 3 algoritmos.

Como parámetros más llamativos para esos algoritmos tenemos, para Elastic Net:

- Alpha: Que sirve de constante multiplicadora para los términos de penalización. En nuestro caso se puede ver que se le pasa una lista con varias alphas.
- Maxiter: Número máximo de iteraciones. En este caso serán unas 5000.

Como parámetros más llamativos para esos algoritmos tenemos, para Gradient Boosting:

- NEstimators: Número de etapas de refuerzo para realizar. En nuestro caso es de 3000. Un número grande generalmente para el Gradient Boosting da buen resultado.
- MaxDepth: Máxima profundidad para estimador regresor individual. En nuestro caso será de 3.
- Loss: En nuestro caso será huber que es una combinación de Is y Lad.
- MinSamplesLeaf: El mínimo número de muestras que se requiere para ser un nodo hoja, en nuestro caso será de 15.
- MinSamplesSplit: El mínimo número de muestras que se requiere para ramificar un nodo interno, en nuestro caso será de 10.

Como parámetros más llamativos para esos algoritmos tenemos, para XGBoost:

- NEstimators: Serán de 7200.
- MaxDepth: Será de 6.

Al final del todo en el script se puede ver como la estimación final es un promedio de las estimaciones de los tres algoritmos de regresión utilizados (Elastic Net, Gradient Boosting y XGBoost).

3. Tabla final de resultados

Fecha	Hora	Posición	Score	Preprocesamiento	Algoritmo	Parámetros	RMSE train
23/12/2017	12:00	1038	0.12657	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies.	Elastic Net	Elastic Net:Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.119214198337
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 minimo muestras para partir nodo con 10.	0.0763571532594
25/12/2017	13:00	1038	0.14136	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies.	Elastic Net	Elastic Net:Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.119214198337
					Random Forest	Gradient Boosting: Estimadores con 4000 maxima profundidad con 4 minimo muestras para partir nodo con 10.	0.13421400424
25/12/2017	16:00	1038	0.14448	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.11921419305625844
					Decision Tree	Decision Tree: Máxima profundidad de 10 Mínimo número de muestras para la partición de un nodo es de 15.	0.18336602233360735
25/12/2017	16:30	1038	0.24298	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.11921418466703167
					Support Vector Regressor	Support Vector Regressor : Número máximo de iteraciones de 3000 Epsilon igual a 0.2 rbf como kernel.	0.17413587923793689
26/12/2017	10:00	935	0.12520	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.119214198337
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.0568918247635
26/12/2017	12:00	935	0.12664	Eliminación características donde más de la mitad de sus valores son NA, llenado de NA, dummies salvo para las características Exterior, en tal caso se hace imputación de valores.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.11924691844805038
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.05525372426994724
26/12/2017	12:30	704	0.12120	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.10919518132276353
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.05330189080647167
26/12/2017	13:00	704	0.12178	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.10917301872807618
					Gradient Boosting	Gradient Boosting: Estimadores con 3500 maxima profundidad con 5 y minimo muestras para partir nodo con 10.	0.051617037891679915
26/12/2017	16:00	688	0.12099	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.1091951814069412
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07375388317842049
26/12/2017	17:00	688	0.12177	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.10897258851186785
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07209852218733985
27/12/2017	19:00	601	0.12051	Eliminación de algunas características donde sus valores son NA, llenado de NA, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.10871744415804033
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07106525465111262
27/12/2017	20:00	601	0.12205	Eliminación de 2 características, imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.1085382484492128
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07153527836510816
29/12/2017	11:00	613	0.12382	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.10853824692912997
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07085867327998868
01/01/2018	11:00	630	0.16925	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.10853824692912997
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.07085867327998868
					Random Forest	Random Forest: Estimadores con 4000 maxima profundidad con 4 y minimo muestras para partir nodo con 10.	0.19136166739402494
02/01/2018	17:00	627	0.12415	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.09833172173468259
02/01/2018	23:00	628	0.12095	Eliminación de 2 características, Eliminación de outliers imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.09833172178187898
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.060417418639638586
03/01/2018	23:50	620	0.12010	Eliminación de outliers Eliminación de outliers previa en train Cambio de variables de tipo cadena a variables de tipo numérico Imputación estudiada de NA, dummies.	Elastic Net	Elastic Net: Alpha desde 0.0001 hasta 10 maximo número de iteraciones igual a 5000.	0.09658446906737084
					Gradient Boosting	Gradient Boosting: Estimadores con 3000 maxima profundidad con 3 y minimo muestras para partir nodo con 10.	0.06073413154465321
					XGBoost	Random Forest: Estimadores con 4000 maxima profundidad con 4 y minimo muestras para partir nodo con 10.	0.04349111990813433

Tabla 3.1: Tabla final del total de subidas a Kaggle para la competión.

Referencias