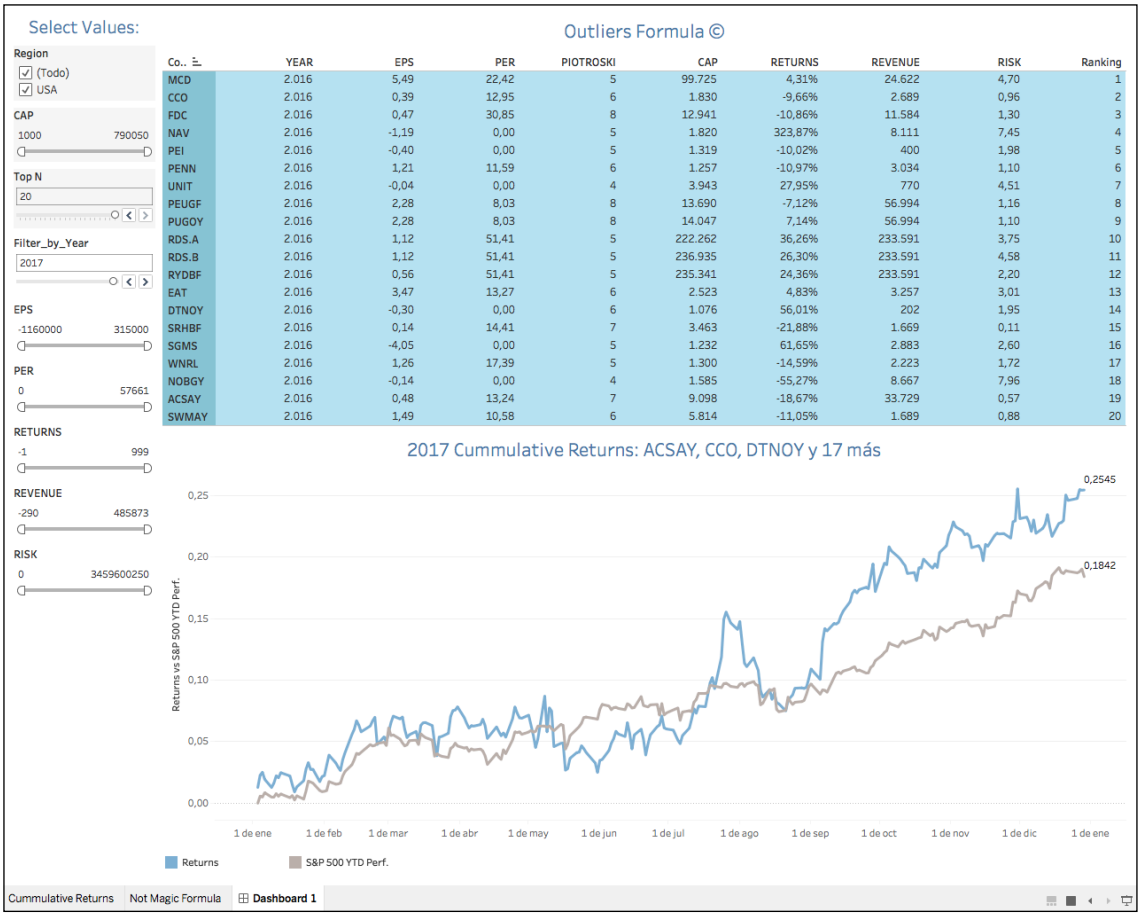
Batiendo a la bolsa

Proyecto Team Outliers - KeepCoding

Belizón, S., Debrán, J., Doutel, F., Lucena, R. Ventas, R. - 22 de noviembre de 2018



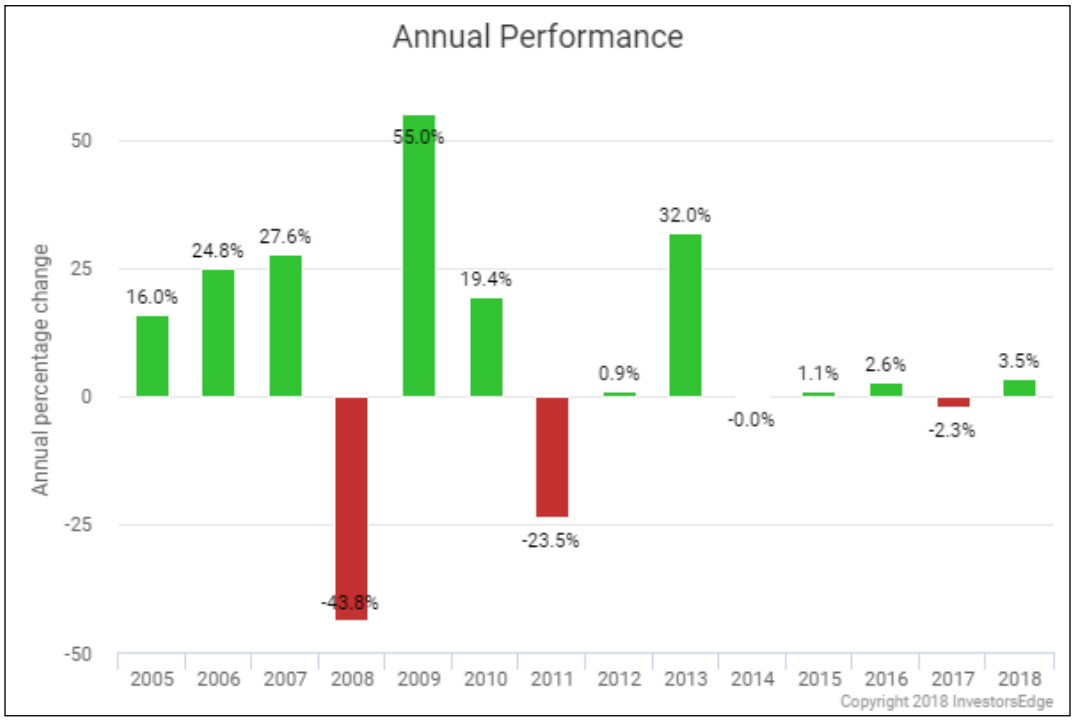
Pitch-Introducción

Batir a la bolsa (ganar más que el índice o una media de empresas) es el objetivo de numerosos analistas e inversores. Generalmente hay dos tendencias, utilizar los valores diarios para hacer una predicción, lo que se denomina **análisis técnico**, o utilizar los indicadores más característicos de las empresas para intentar elegir cuáles van a ser las empresas más rentables, lo que se conoce como **análisis fundamental**.

El objetivo de nuestro equipo, **Outliers**, es utilizar los conocimientos que hemos adquirido a lo largo del bootcamp para batir al índice S&P 500 de Estados Unidos mediante análisis fundamental.

Hemos partido de la **fórmula mágica de Joel Greenblatt**, un famoso inversor y académico norteamericano, el cual consiguió batir al índice utilizando tan solo dos indicadores (ROIC o Retorno sobre el capital invertido, que muestra lo rentable que es una empresa en función de lo que se ha invertido en ella, y EV/EBIT, indicador fundamental que permite conocer lo barata o cara que se encuentra una compañía). La fórmula de Joel Greenblatt ordena a modo de ranking y de forma anual el universo de empresas sobre el que se trabaja para determinar en qué empresas invertir el año siguiente entre el 1 de enero y el 31 de diciembre.

De acuerdo con algunos autores, aplicando dicha fórmula la **rentabilidad anual media estaría entre un 20% y un 24% anualizado entre 1988 y 2009**. Aunque dicha fórmula ha dejado de funcionar en los últimos 8 años debido a un aumento de la eficiencia del mercado respecto a este método de inversión y en años de crisis financieras (2008, 2011) ha arrojado malos resultados, sirve como inspiración y punto de partida para nuestro proyecto.



Rendimiento anual de acuerdo a la fórmula mágica de Greenblatt

¿Podríamos utilizar técnicas de ML o DL para determinar nuevos indicadores que superen al índice S&P 500? Hemos aplicado los conocimientos adquiridos en los distintos **módulos del bootcamp Big Data y Machine Learning de Keepcoding** para ponernos a prueba. Así, hemos utilizado las competencias adquiridas de **Big Data y SQL avanzado** para la obtención y puesta a punto del dataset, **Arquitectura** para ver en qué entorno deberíamos realizar nuestro proyecto, **Análisis exploratorio y Data Mining** para sacar el jugo a nuestros datos, **Machine Learning** y **Deep Learning** para el entrenamiento de modelos y la extracción de resultados, así como visualización para el análisis interactivo y presentación en **Tableau**, …. sin olvidar las competencias adquiridas en **Álgebra Lineal y Estadística** como base para comprender los módulos anteriores.

Dicen que el camino es más importante que la meta. En este proyecto hemos invertido muchísimas horas de esfuerzo para cumplir un objetivo. También ha sido una buena oportunidad para seguir aprendiendo y para poner en práctica todas nuestras competencias, así como para trabajar en equipo mediante la metodología Scrum.

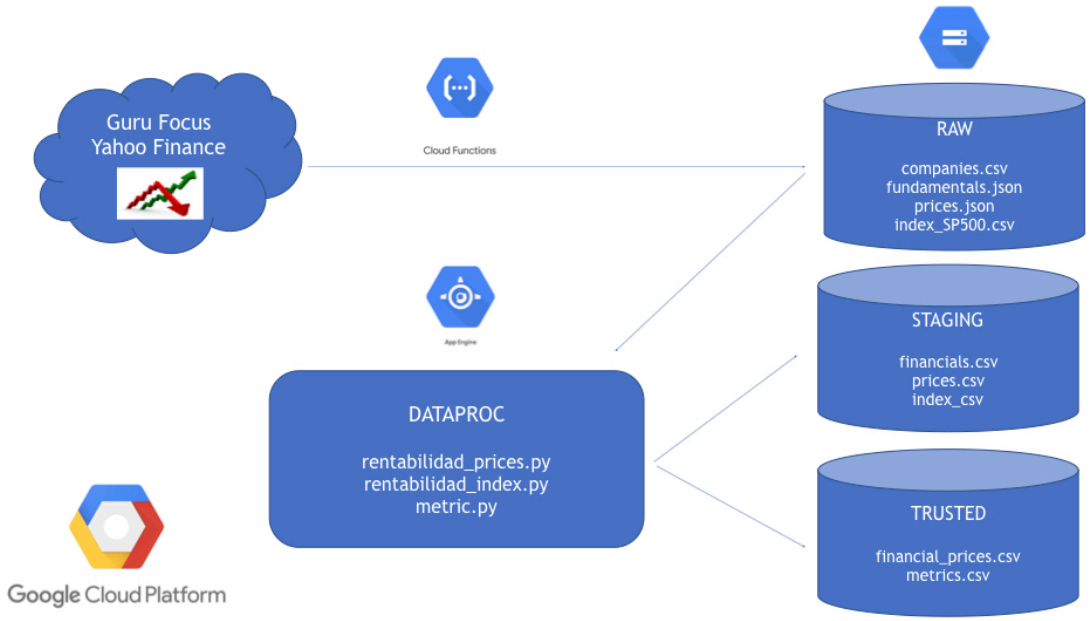
Descripción del proyecto

Arquitectura

Decidimos montar la **arquitectura en Google Cloud**, ya que contábamos con créditos gratuitos. Para la primera parte, la de extracción de datos mediante *scraping* y llamadas a APIs, usamos *cloud functions*, que nos permiten ejecutar una arquitectura *serverless* de bajo coste. Los datos resultantes los volcamos en la **zona RAW** de nuestro *bucket*. Teníamos dos GB de información, así que decidimos proceder al tratamiento de la misma con Jobs de pySpark, ejecutados en Google Dataproc.

Los archivos JSON no venían en formato legible para Big Query, Pandas o Spark ya que uno venía anidados, …, por lo que hubo que ponerlos en un formato procesable.

En esta parte usamos dos Jobs (rentabilidad_prices.py y rentabilidad_index.py) que se encargaban de procesar por un lado el fichero prices.json para añadir las rentabilidades diarias y acumuladas, y por otro lado el fichero index.csv para añadir también las rentabilidades diarias y acumuladas del índice. El resultado de estos Jobs son dos csv que se guardan en la **zona de STAGING**. Por último unimos los ficheros con los datos financieros y los datos de los precios y formamos el fichero financial_prices.csv. También hicimos el tratamiento de las métricas necesarias para la prueba de Greenblatt en Tableau, generando el fichero metrics.csv. Todo esto quedó guardado en la **zona de TRUSTED**, que será nuestro punto de partida para continuar con el análisis de datos.



También hemos generado archivos trimestrales en local y realizado otras pruebas a la vez que íbamos montando nuestra arquitectura.

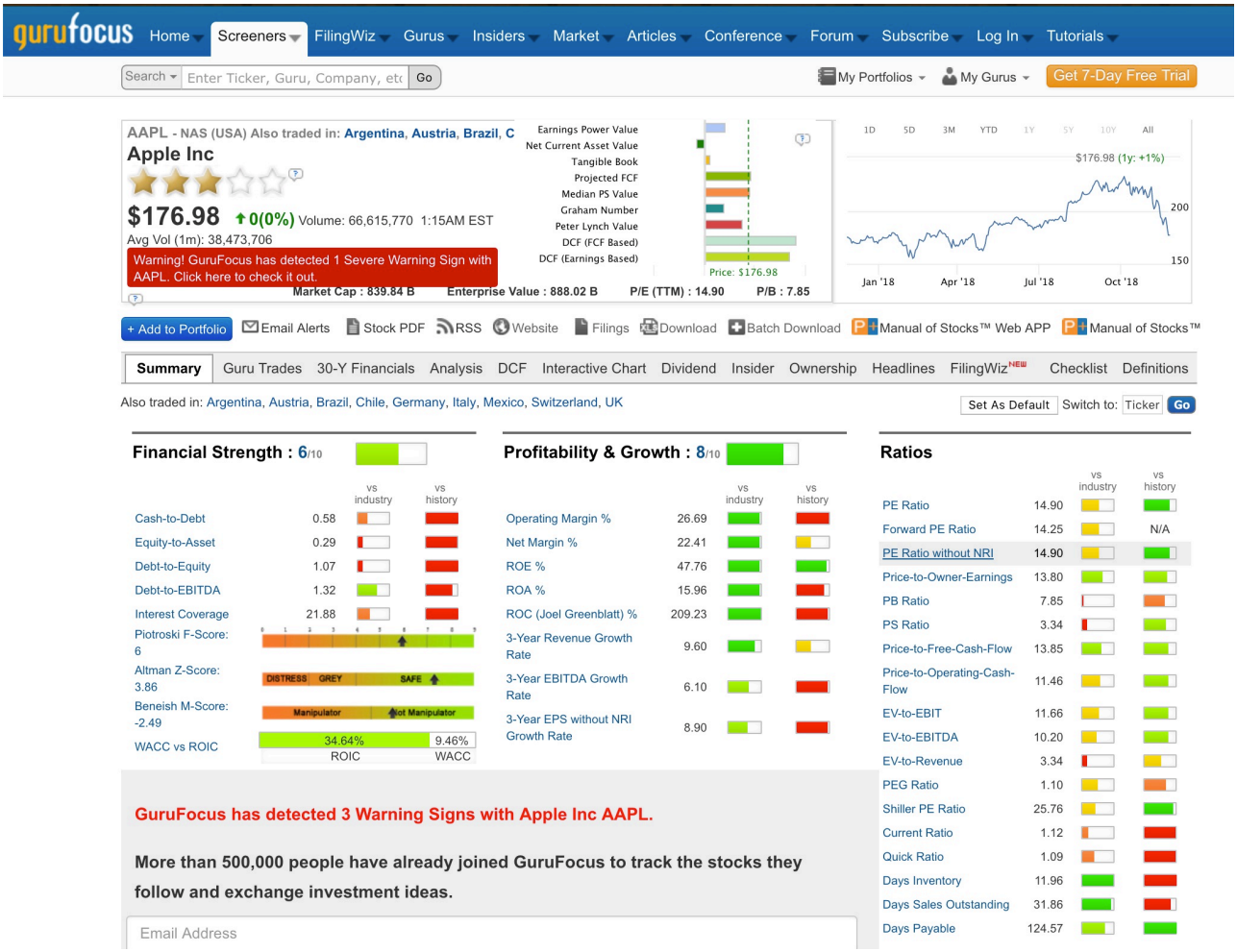
La visualización en Tableau la hemos construido a partir de una tabla en BigQuery alimentada por los archivos .csv mencionados anteriormente y una conexión directa desde Tableau.

La justificación de por qué hemos elegido esta arquitectura no sólo se basa en la obtención de los dataset, sino dejar preparado para el futuro la obtención de archivos incrementales y que los datos se actualizasen periódicamente. Así podríamos aplicar directamente los algoritmos de Machine Learning en el Cloud y volcar los datos directamente en Big Query para que Tableau se actualizase automáticamente y **productificar nuestra solución**.

Scraping

Para batir al mercado bursátil norteamericano necesitamos datos. Durante el **primer sprint** nos dedicamos a obtener mediante técnicas de *scraping* los datos financieros de **20.000** empresas entre los años **1970 y 2017**, para al final quedarnos con **9.982 compañías**.

El *scraping* no resultó ser ni fácil ni mucho menos sencillo. Fue necesario preparar código que fuera extrayendo los datos del sitio web Gurufocus (recurso web de prestigio para la realización de estudios fundamentales de empresas a nivel mundial) utilizando las APIs que proporciona para la extracción de los datos.

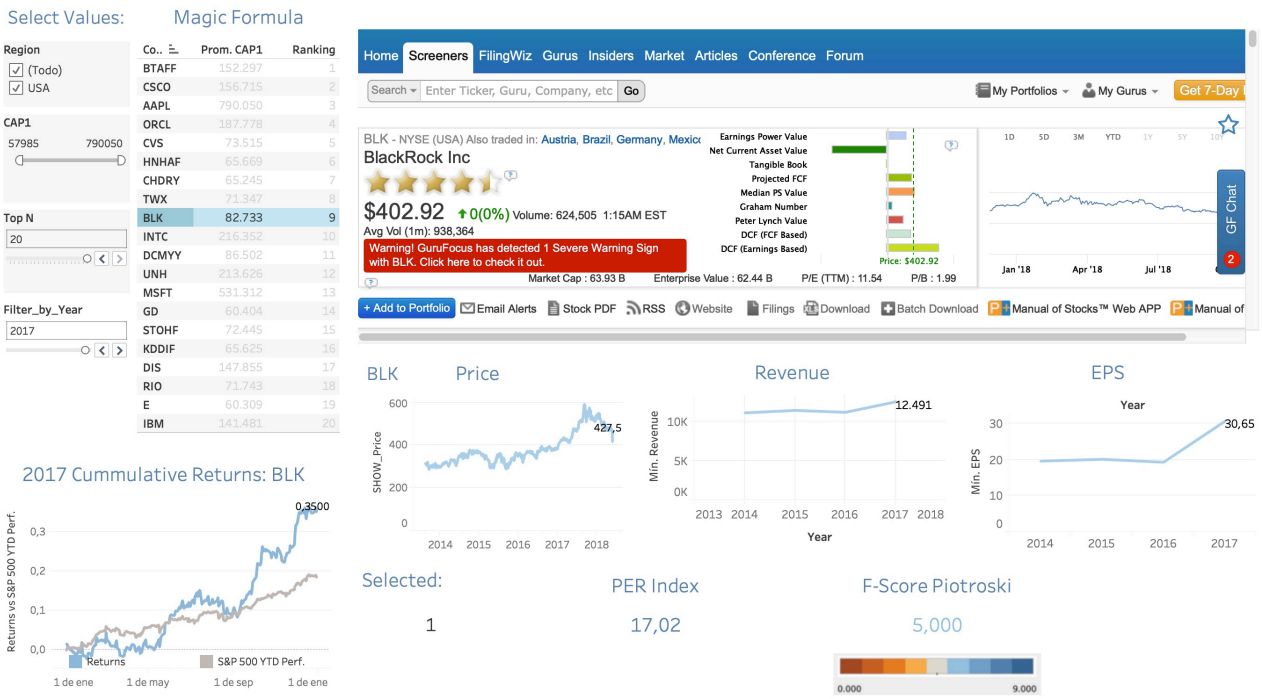


Captura de Gurufocus

Para la obtención de los datos usamos por un lado *scraping* y por otro la API que nos proporciona el sitio web de Gurufocus. En un primer momento "escrapeamos" la página web para obtener los códigos de las empresas. Con estos códigos hicimos llamadas a la API de Gurufocus, una para obtener los datos fundamentales y otra para obtener los

precios. Por otro lado, fue necesario también realizar un *scraping* adicional para obtener los sectores a los que pertenecían las compañías. Fueron necesarios **diversos *scraping* para hacernos con los datos**. Los datos de la API nos venían en JSON anidados, por lo que fue necesario "aplanarlos" para proceder a su tratamiento con la librería pandas.

En paralelo con la actividad de *scraping* decidimos hacer **una prueba de la fórmula mágica de Greenblatt en Tableau** para ver si podíamos replicar los resultados con un subconjunto de empresas. Así, con un dataset reducido y convenientemente procesado para la obtención de los indicadores de la fórmula y sus correspondientes ranking nos llevamos dichos datos a Tableau y comprobamos que bajo las condiciones de Greenblatt se batía al índice (salvo las excepciones mencionadas en la introducción de este documento).



Nuestra prueba de análisis para batir a Greenblatt en Tableau

Pese a que Gurufocus es un sitio de pago y, por lo tanto, los datasets no son gratuitos, vimos que para nuestra desgracia la calidad de los datos no era la adecuada. Así, numerosas empresas contaban con NaNs en sus datos, valores a cero, muchas tenían reflejados pocos años de datos… **La calidad del dato ha sido un verdadero quebradero de cabeza,** y nos imaginamos que en numerosos proyectos será así también. Decidimos seguir buscando otras alternativas como fuentes de datos y encontramos un sitio web (Simfin). De ahí conseguimos descargar otro dataset. Después del pre - procesado nos dimos cuenta de que perdíamos bastante información (muchas menos empresas, menos columnas, …), por lo que volvimos a darle otra oportunidad al dataset de Gurufocus. Para obtener mayor cantidad de datos hicimos el procesamiento (aplanamiento de los JSON y

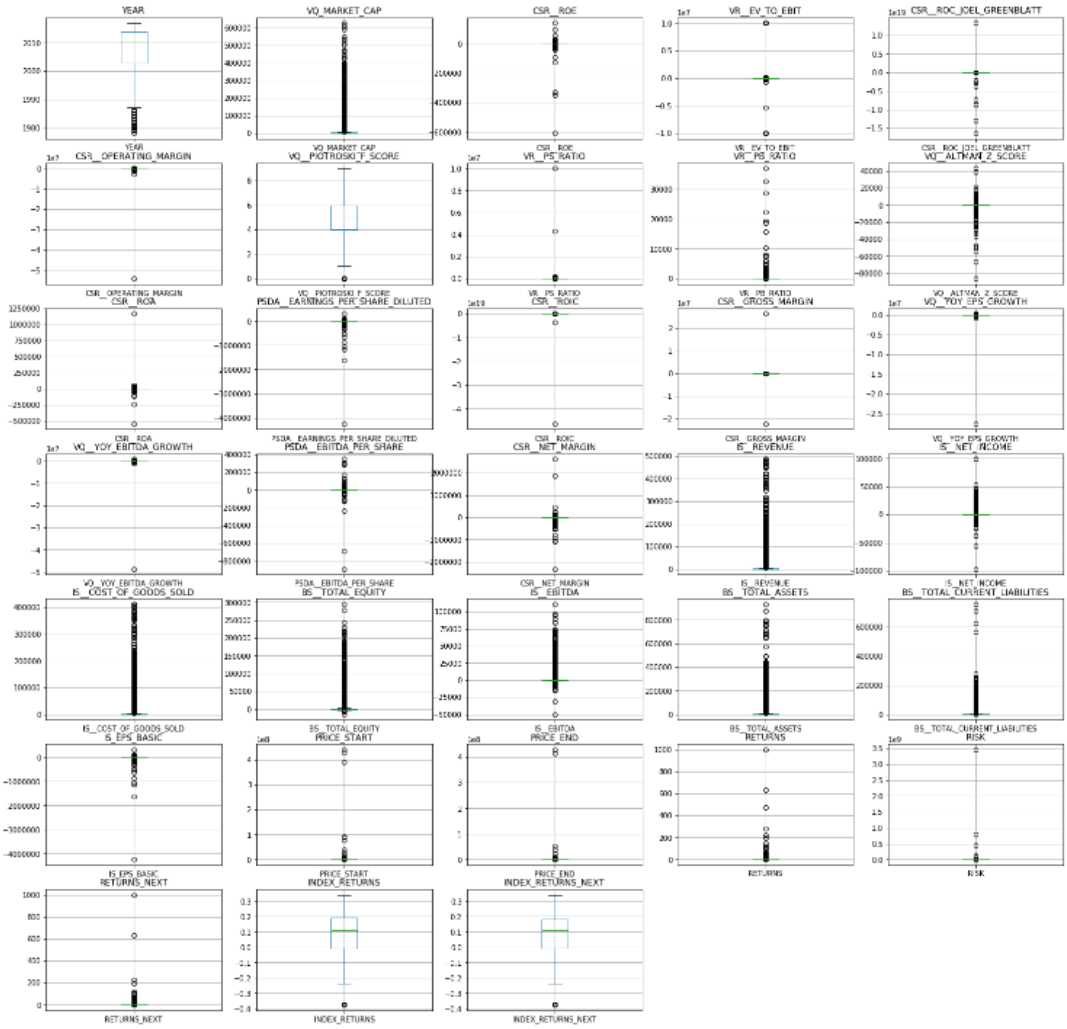
cruce con prices) y nos quedamos al final con **datos trimestrales**, que parecían tener mayor calidad.

Todo el procesado de los datos nos ha llevado mucho tiempo y esfuerzo para tener un dataset con el que poder probar y extraer información.

Análisis Exploratorio

Uno de los problemas que descubrimos al hacer el análisis exploratorio fue que el año que reflejaba Gurufocus coincidía con la **fecha de presentación de resultados y no con el año fiscal** al que se referían los mismos. Por esta razón y por las detalladas con anterioridad decidimos continuar con los datos de Simfin, que garantizaban que este dato era el correcto. Finalmente hemos preparado **dos datasets: uno con datos anuales y otro con datos trimestrales** y realizado pruebas con ambos.

Dada la baja calidad de los datos del dataset, seleccionamos una serie de características financieramente más significativas y las estudiamos.



Datos anuales, estudio de características y outliers

Datos anuales, histograma

Tras eliminar los valores nulos y los outliers, nuestro dataset se ve drásticamente reducido, pasando de **87.438 a 5.421** filas.

Además, tras preparar nuestra target dicotómica, vemos que **ambas clases se superponen para todas las variables** y, al no ser linealmente separables, hacen que la resolución del problema planteado se nos antoje bastante difícil.

Datos anuales, vemos una superposición de las variables, lo cual dificulta la separación categórica de nuestra variable objetivo

En paralelo y cuando estuvo disponible el dataset, realizamos un análisis exploratorio de los datos trimestrales. Primero unificamos los rangos de las fechas y vimos que no todas las empresas presentan resultados el mismo trimestre. Tras una observación más exhaustiva nos dimos cuenta que cuando más había era en los periodos comprendidos entre marzo y junio, septiembre y diciembre. Luego borramos aquellas columnas que tenían más de un 10% de valores a NaNs. También aquellas que venían con más de un 10% de los valores a cero. Otra cosa que tuvimos en cuenta fue el campo

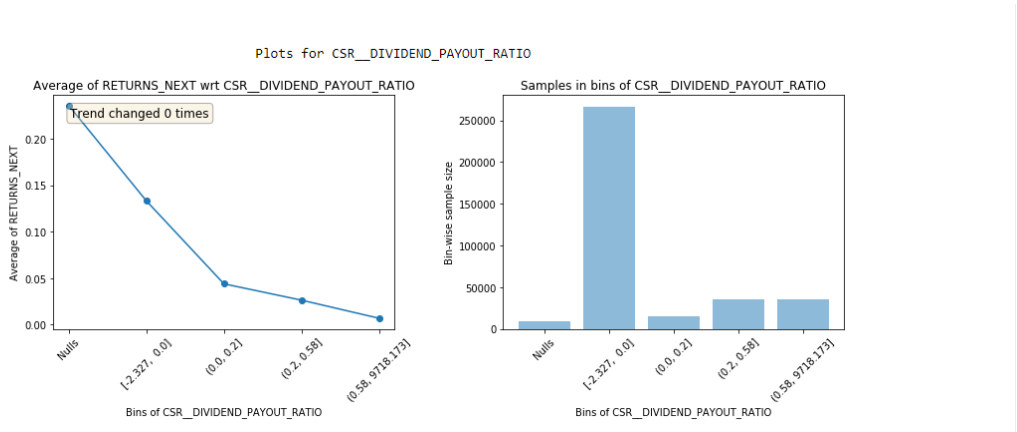
PRELIMINARY, que indicaba a 1 si los valores no eran definitivos ó 0 en caso contrario, por lo que procedimos a eliminar todas las filas con PRELIMINARY igual a 1.

Después vimos con la matriz de correlación que había **muchas columnas fuertemente correladas**. Decidimos eliminar aquellas cuyo valor absoluto de correlación era superior a un umbral, que fijamos en 0.9. Otra cosa que nos pareció de interés es mirar si existía correlación entre la variable target y el resto, observando que no era el caso.

También decidimos quedarnos con empresas de más capitalización, ya que los datos para este tipo de empresas suelen ser más estables. Como el dato de capitalización viene dado por años, borramos aquellas empresas que tienen un valor de capitalización media pequeño en todos los años del estudio. El valor de capitalización viene en millones de dólares, y por tanto seleccionamos empresas con una **capitalización mayor del millón de dólares**. En total nos quedamos con **7.638 empresas**.

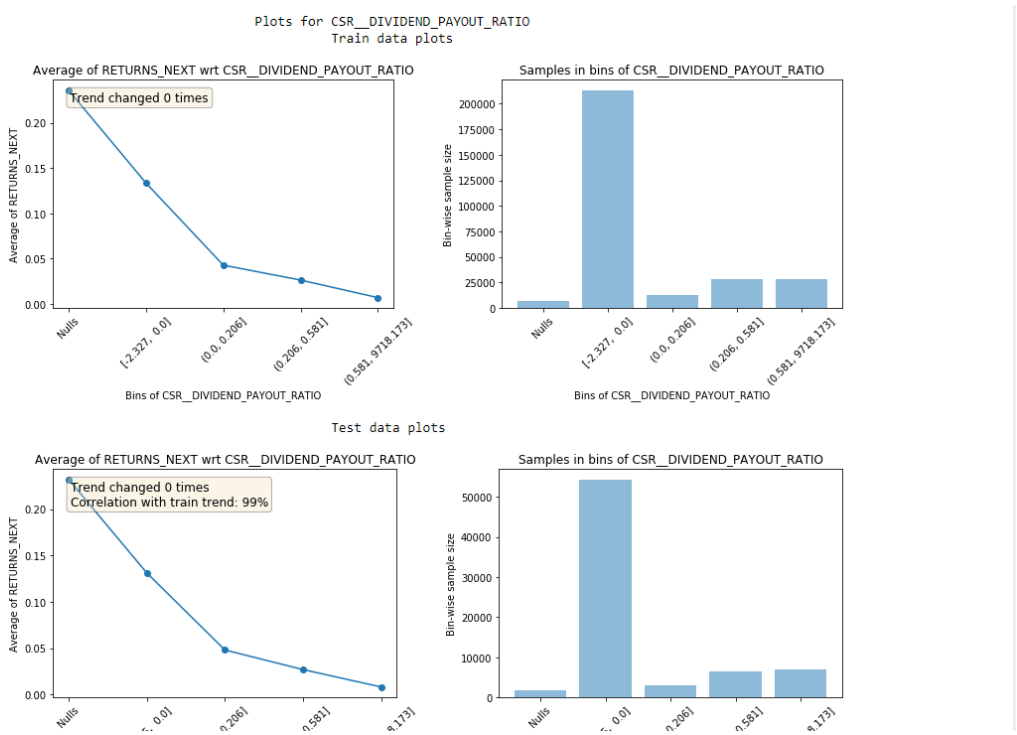
Después de esta primera depuración en los datos, y usando la librería featexp, llevamos a cabo una serie de gráficos para intentar comprender mejor las variables que tenemos y también identificar aquellas que son ruidosas.

Para poder comprender las variables, la librería proporciona unos gráficos que dividen la población en partes iguales y muestra la media de la variable target para cada parte:



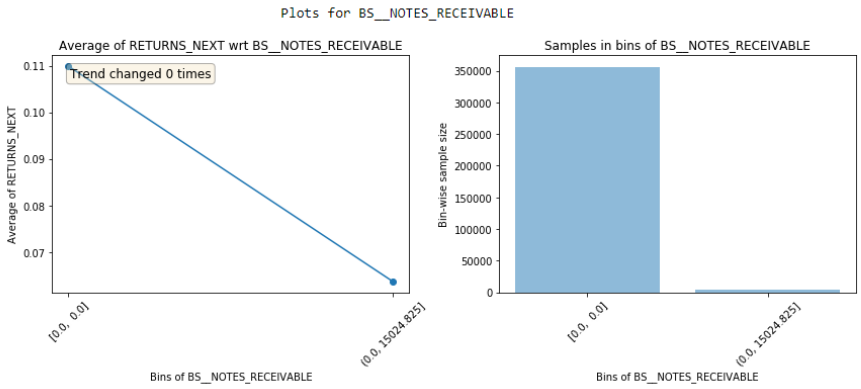
Ejemplo de gráficos para comprender las características mostrando la media de la variable target

Para la identificación de variables ruidosas, dividimos los datos en dos grupos (train y test) y la librería nos muestra cómo se distribuyen los datos en cada uno de los grupos para ver si los valores se comportan igual en train y test.



Vemos que la variable mostrada no presenta ruido, por lo que no la descartamos

Después de este estudio decidimos eliminar las variables ruidosas y aquellas que no aportan ninguna información porque tienen gran parte de la población en un valor constante:

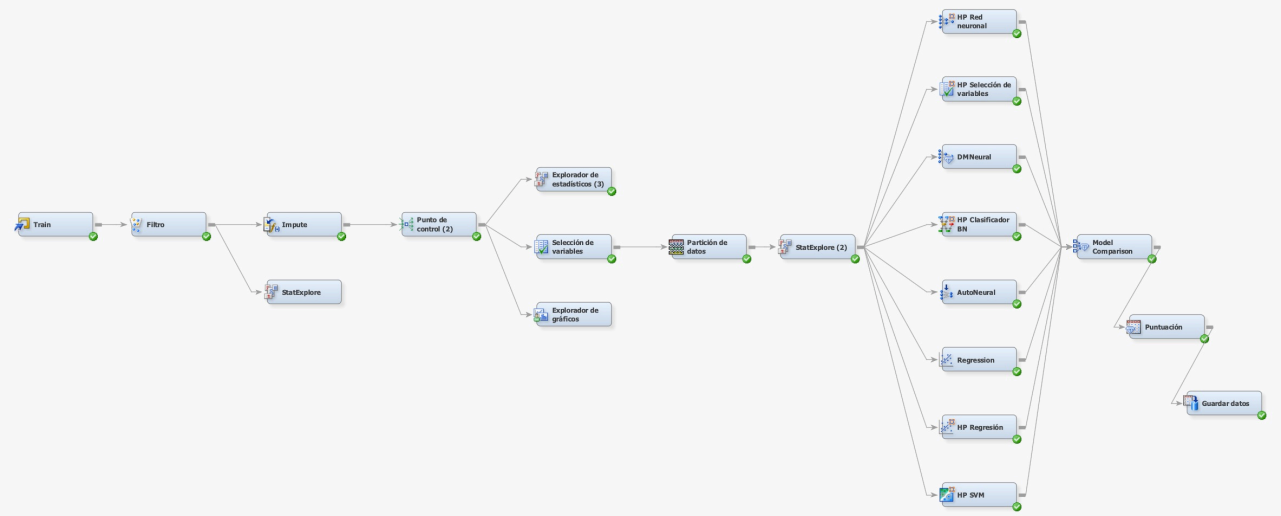


Ejemplo de variable que hemos descartado, ya que tiene muchos valores a cero

Tras quedarnos con las columnas más relevantes, procedimos al tratamiento de outliers y NaNs (asignando la media a dichos valores) para continuar con nuestro análisis.

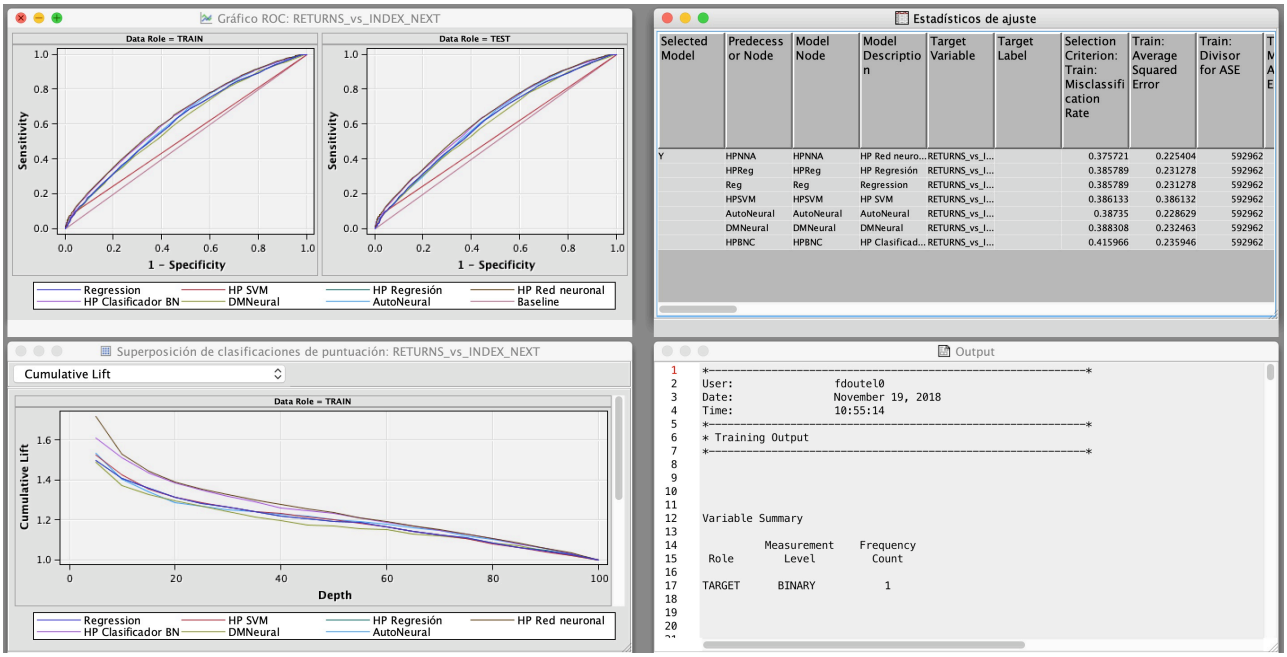
Data Mining con SAS

Hemos utilizado SAS Enterprise Miner para la evaluación de diferentes técnicas exploratorias y determinar si alguna de ellas arrojaba buenos resultados. La hipótesis a comprobar era si podíamos **predecir que las empresas iban a superar al índice S&P 500 en el periodo siguiente** utilizando los datos de análisis fundamental. Hemos tratado todas las empresas de forma análoga y no hemos hecho distinciones por periodo. Hemos utilizado los años entre 2010 y 2016 y también entre 2000 y 2016 y hemos hecho una división 80/20 para el entrenamiento.



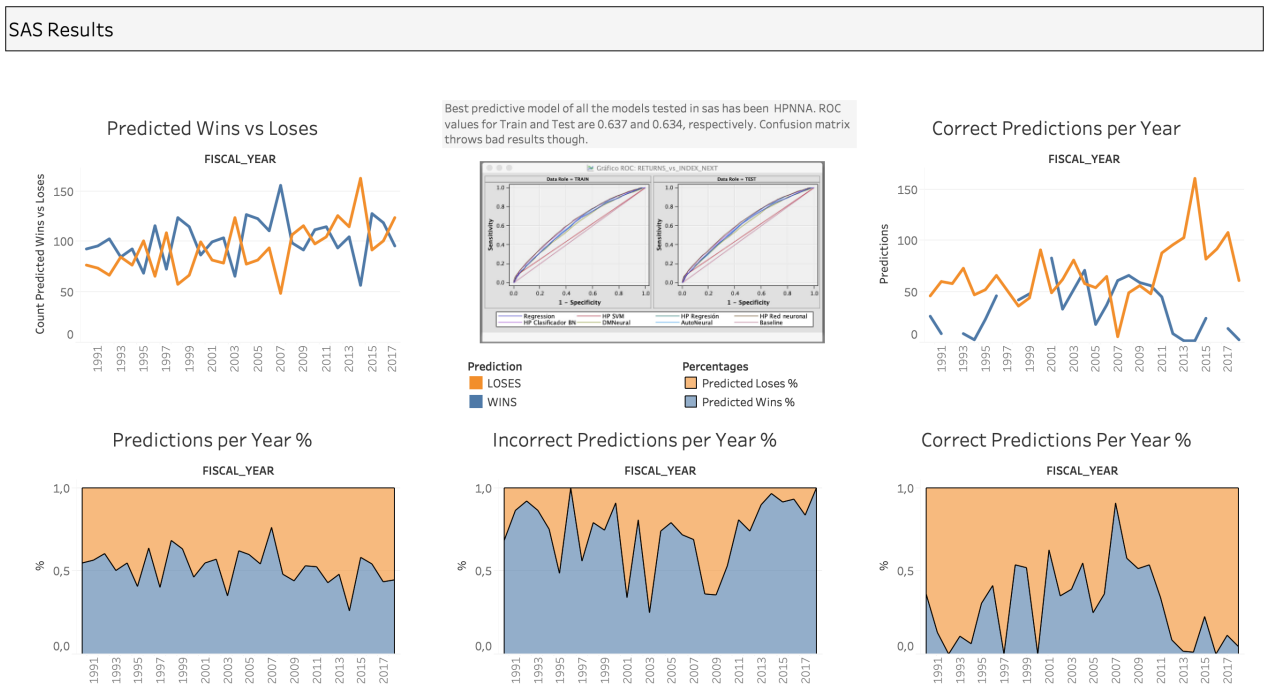
Estructura de nuestro estudio en SAS Enterprise Miner

Una vez estudiados diversos métodos de ML y DL y analizados los resultados, **el mejor método de acuerdo con SAS Miner es Análisis Neural HP**, pero viendo la matriz de confusión vemos que la clasificación es buena para los casos en los que no se bate al índice, y mala para el caso de superarlo, lo cual va en detrimento de lo que querríamos obtener.



Resultados arrojados por SAS Enterprise Miner

No obstante, nos hemos llevado estos datos a Tableau para ver si podríamos extraer alguna conclusión de los mismos. No olvidemos que nuestro objetivo es batir al mercado y, teniendo en cuenta los datos, obtener unos buenos resultados en el análisis no vemos que pueda ser factible.



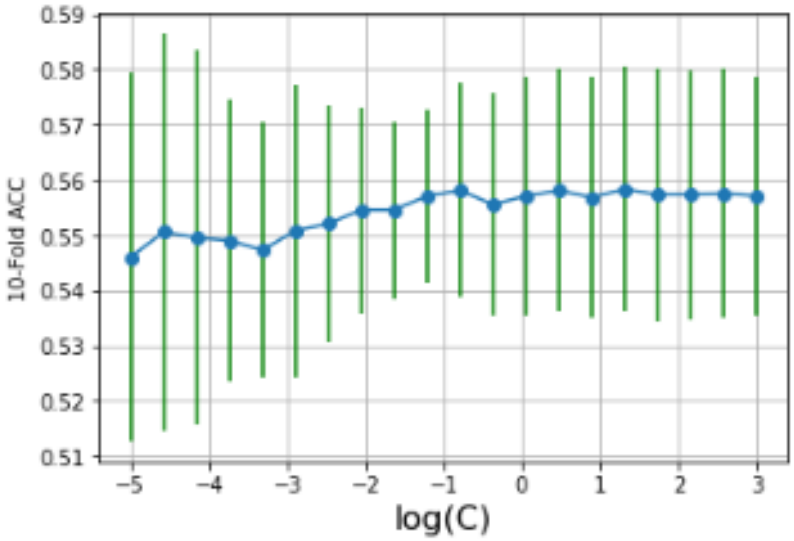
Vemos muy pocas predicciones correctas en general en Tableau

En la parte de Machine Learning hemos abordado el problema desde varios enfoques. Por un lado, lo hemos planteado como un problema de **clasificación binaria**, en el que la variable a predecir es si se bate o no al índice.

En el caso de los datos anuales y para el problema de clasificación, obtuvimos unos gráficos que nos indicaban que el problema iba a ser difícil de resolver, tal y como se detalló en el análisis exploratorio. Pese a todo, nos aventuramos a **tratar de obtener el mejor resultado posible**, aplicando varios modelos predictivos estudiados durante el Bootcamp.

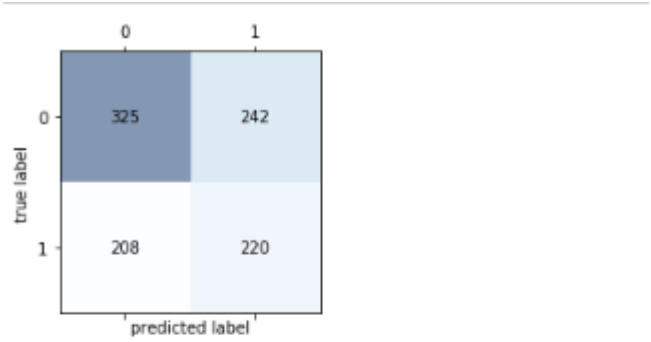
Primeramente, probamos con una simple **regresión logística**, cuya optimización da como resultado un score de 0.55:

```
best mean cross-validation score: 0.558
best parameters: {'C': 20.6913808111479}
```

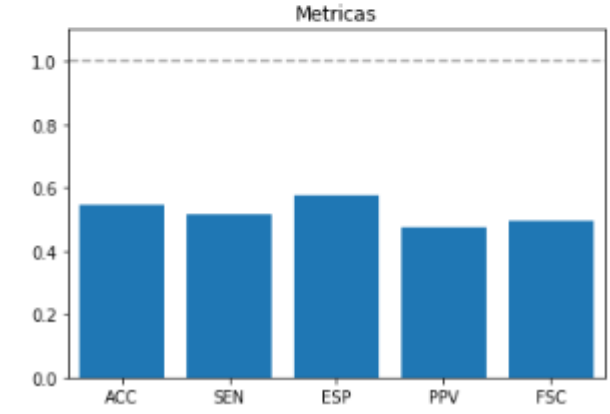


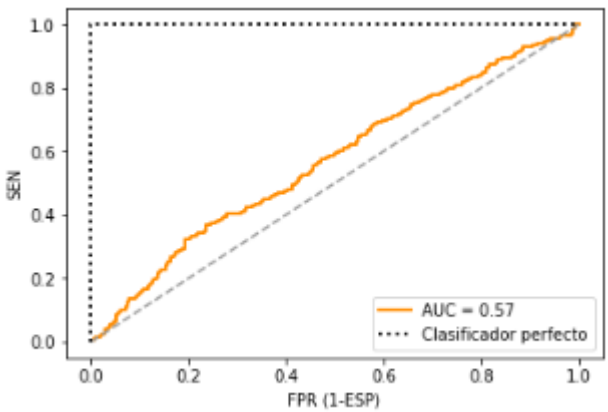
Resultados de la regresión logística

Y, tras realizar una predicción sobre nuestros datos de test para el año 2.016, obtenemos las métricas y curva ROC siguientes:




```
ACC:  0.5477386934673367
SEN:  0.514018691588785
ESP:  0.5731922398589065
PPV:  0.47619047619047616
FSC:  0.49438202247191004
```

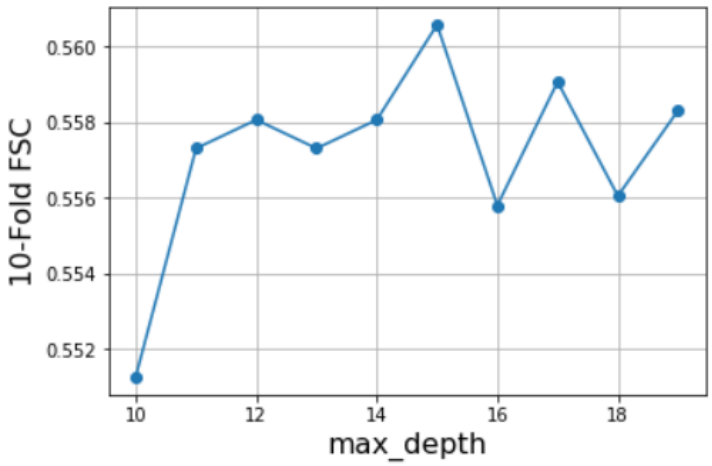




Métricas y curva ROC obtenidas

Luego probamos con los **árboles de decisión**. Hicimos cross-validation para obtener los parámetros óptimos, pero al comprobar las prestaciones de los modelos vimos que se producía overfitting, por lo que tuvimos que seleccionar menos profundidad para reducir la complejidad del modelo.

```
best mean cross-validation score: 0.561
best parameters: {'max_depth': 15}
```



Resultados de aplicar árboles de decisión

```python
depthOpt = grid.best_params_['max_depth']

rf = RandomForestClassifier(random_state = 0,
                            #max_depth = depthOpt,
                            max_depth = 5,
                            n_estimators = 200,
                            max_features = 'sqrt').fit(X_train, y_train)
ytrainRandomForest = rf.predict(X_train)
ytestRandomForest  = rf.predict(X_test)

mseTrainModelRandomForest = mean_squared_error(y_train, ytrainRandomForest)
mseTestModelRandomForest = mean_squared_error(y_test, ytestRandomForest)

print('MSE Modelo Random Forest (train): %0.3g' % mseTrainModelRandomForest)
print('MSE Modelo Random Forest (test) : %0.3g' % mseTestModelRandomForest)
print("R^2 (train): {:.4f}".format(rf.score(X_train, y_train)))
print("R^2 (test): {:.4f}".format(rf.score(X_test, y_test)))
```

```
MSE Modelo Random Forest (train): 0.362
MSE Modelo Random Forest (test) : 0.414
R^2 (train): 0.6378
R^2 (test): 0.5859
```

Utilizando un clasificador Random Forest

Otro algoritmo que hemos probado son los **Gradient Boosted Trees**, que al igual que en el caso anterior con los parámetros óptimos también tenía overfitting, por lo que tuvimos que cambiarlos para reducirlo.

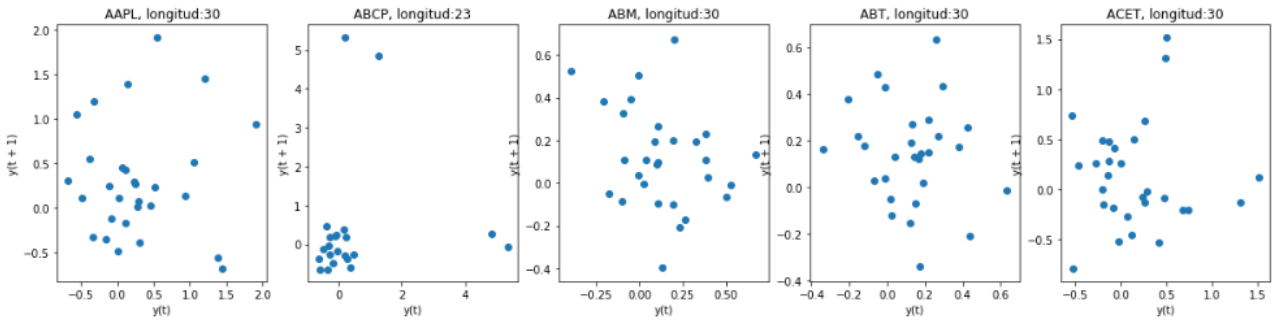
```
learningRateOpt = grid.best_params_['learning_rate']
nEstimatorsOpt = grid.best_params_['n_estimators']

bf = GradientBoostingClassifier(random_state = 0,
                                max_depth = 5,
                                #learning_rate = learningRateOpt,
                                #n_estimators = nEstimatorsOpt,
                                learning_rate = 0.01,
                                n_estimators = 150,
                                max_features = 'sqrt').fit(X_train, y_train)
ytrainBoostingForest = bf.predict(X_train)
ytestBoostingForest  = bf.predict(X_test)

mseTrainModelBoostingForest = mean_squared_error(y_train, ytrainBoostingForest)
mseTestModelBoostingForest = mean_squared_error(y_test, ytestBoostingForest)

print('MSE Modelo Boosting Forest (train): %0.3g' % mseTrainModelBoostingForest)
print('MSE Modelo Boosting Forest (test) : %0.3g' % mseTestModelBoostingForest)
print("R^2 (train): {:.4f}".format(bf.score(X_train, y_train)))
print("R^2 (test): {:.4f}".format(bf.score(X_test, y_test)))
```

```
MSE Modelo Boosting Forest (train): 0.321
MSE Modelo Boosting Forest (test) : 0.415
R^2 (train): 0.6795
R^2 (test): 0.5849
```

Gradient Boosted Trees, resultados

Por otro lado, para conocer la rentabilidad de una empresa (y, por tanto, si batirá o no al índice), debemos hacer el estudio por empresa. Para esto usamos **técnicas de regresión y algoritmos de series temporales**.

Para abordar la parte de series temporales, empezamos haciendo un estudio de la posible **correlación de la variable target t con el valor en t+1** (test de autocorrelación), comprobando que no era así para la mayoría de las empresas:



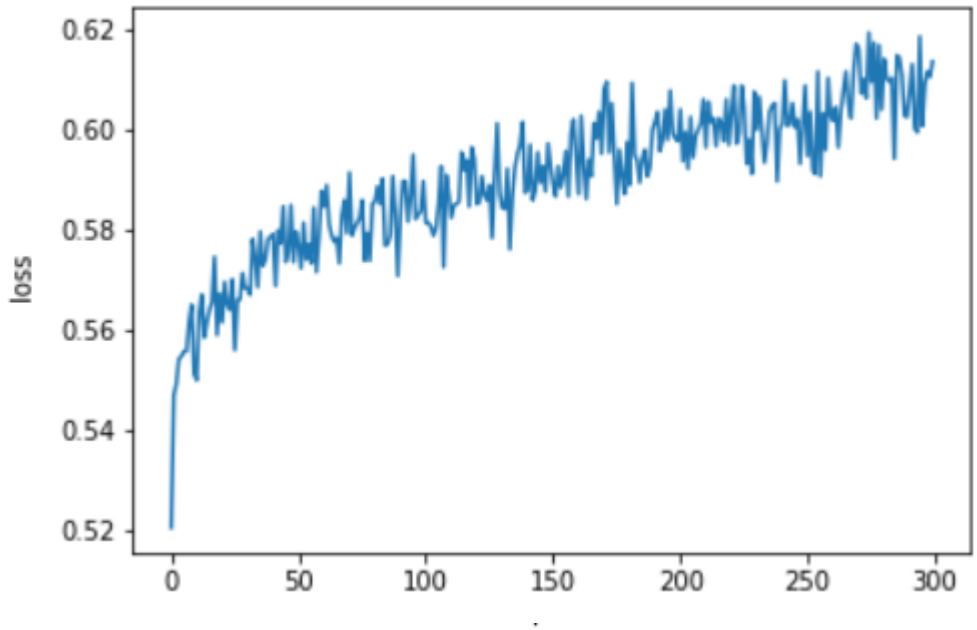
Estudio de correlación entre la variable target en t y en t+1

Por ello optamos por un algoritmo de predicción de series temporales multivariable, para lo que usamos la implementación del **algoritmo VAR** (Vector autoregresion) de statsmodels.tsa.vector_ar.var_model. Para medir el performance del modelo establecimos un modelo base, en el que simplemente decimos que la rentabilidad de una empresa en t+1 será la misma que en t. Luego abordamos el problema de dos formas diferentes. Primero con una **división por empresa con datos anuales**, obteniendo resultados muy malos, que no conseguían mejorar los errores cuadráticos medios del modelo base. Optamos (una vez obtenidos los datos trimestrales) por otra aproximación, que consistía en **coger los datos trimestrales, juntarlos con los precios mensuales e interpolar los valores NaNs.** Así conseguíamos multiplicar por 12 los datos.

In [40]:
```
cols = dfApple.columns
#converting predictions to dataframe
pred = pd.DataFrame(index=range(0,len(p)),columns=[cols])
for j in range(0,len(cols)):
    for i in range(0, len(p)):
        pred.iloc[i][j] = p[i][j]
    #check rmse
try:
    rmse_var = sqrt(mean_squared_error(pred['RETURNS_prices'], valid['RETURNS_prices']))
    print('rmse(VAR) value for RETURNS is:', rmse_var, 'vs rmse(base)',rmse)
except:
    pass
```

```
rmse(VAR) value for RETURNS is: 0.6626339042932788 vs rmse(base) 0.015892512372916223
```

Aun así, seguían siendo insuficientes y los resultados igual de malos que con los datos anuales.

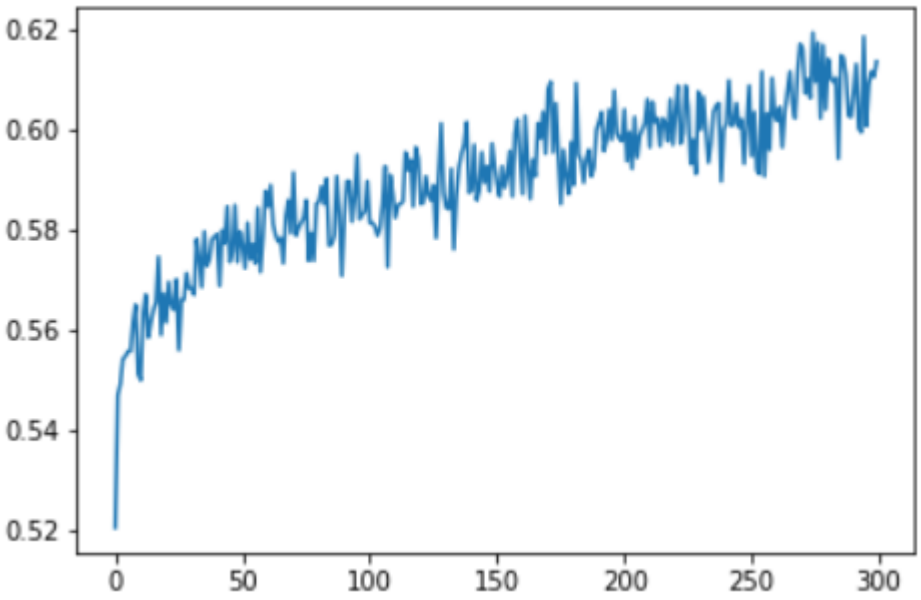


Deep Learning

Para la resolución de nuestro problema mediante la utilización de redes neuronales, hemos trabajado con distintos modelos. Al principio, intentamos resolver el problema desde un punto de vista global, es decir, introduciendo todos los parámetros fundamentales que teníamos de las empresas **sin tener en cuenta la temporalidad**. Las primeras pruebas utilizaban un modelo de regresión para intentar predecir la rentabilidad del año siguiente. Esto nos arrojó unos muy malos resultados (accuracy muy baja), y tras numerosas pruebas entendimos que éste no era el camino a seguir, ya que no disponíamos ni de suficiente cantidad de datos, ni estos tenían la calidad adecuada como para poder realizar una predicción de un número tan preciso.

A continuación, se enfrentó el problema con un **modelo de clasificación**, el cual debería ser capaz de indicar si una empresa con determinados parámetros fundamentales en el año n, sería capaz o no de batir al índice en el año n+1. Con este modelo tuvimos una importante mejora en los resultados, llegando a un **accuracy de un 57,2%**. Si bien el resultado era mejor, las predicciones que obteníamos a la hora de seleccionar empresas no mostraban un rendimiento adecuado a nivel de lógica de inversión.

A pesar de que intentamos "tunear" el modelo para mejorar sus resultados, no fuimos capaces de obtener predicciones adecuadas.



Vemos que el entrenamiento no es bueno

```
# Evaluamos el modelo
model.evaluate(X_test, y_test)
```

```
995/995 [==============================] - 0s 151us/step

[0.6782614367691117, 0.5728643216679443]
```

Resultados de Train y Test

Llegados a este punto, decidimos **intentar tener en cuenta la temporalidad**. Para ello, utilizamos un dataset que tuviera datos fundamentales pero de forma trimestral. Separamos los datos por empresa, pensando que aumentaría la capacidad predictiva del modelo. Desafortunadamente, y tras numerosas pruebas con un modelo LSTM, concluimos que la capacidad de predicción también era inadecuada para nuestro problema.

Consideramos que, para la correcta utilización de modelos de Deep Learning en la resolución de problemas como el nuestro, **es necesaria una cantidad mucho mayor de datos y, sobretodo, que la calidad de los mismos sea alta**.

Resolviendo el problema

Una vez realizadas las partes correspondientes a Machine Learning, Deep Learning y SAS, y teniendo en cuenta que con el volumen y la calidad de los datos no conseguíamos resolver nuestro reto de forma pura por ninguno de ellos, **tuvimos que volver al origen**. Lo que siempre hemos querido es resolver un criterio de inversión financiera, no un problema puro de clasificación, aprendizaje neuronal o aplicar modelos de SAS.

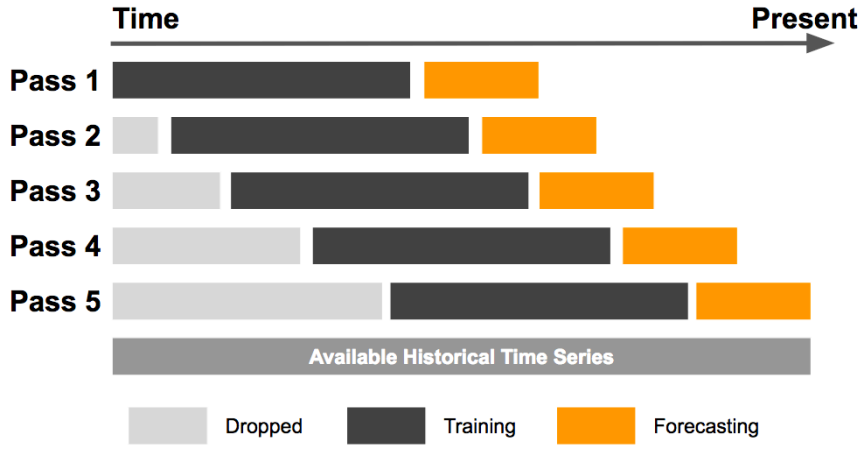
Para la resolución se aplican conocimientos financieros así como otra información relevante que pueda modular los resultados. Greenblatt cogía dos indicadores, pero los ordenaba por sus ranking y cogía los mejores valores para elegir empresas para invertir. Volviendo al concepto de la fórmula mágica, pensamos que podríamos **obtener unos coeficientes mejores que el propio Greenblatt,** y no dar el mismo peso a los conceptos de ROIC o Retorno sobre el capital invertido, que muestra lo rentable que es una empresa en función de lo que se ha invertido en ella, y EV/EBIT, indicador fundamental que permite conocer lo barata o cara que se encuentra una compañía. Así, empíricamente y por medio de una regresión vimos que tenía mucho más sentido dar un peso de 0.34 a la ROIC y 0.28 al EV/EBIT.

```
MSE Modelo Lineal Simple (train): 999489.3
MSE Modelo Lineal Simple (test) : 1000655
R^2 (train): 0.2283
R^2 (test): 0.2338
Coefficients:
CSR__ROC_JOEL_GREENBLATT_RANKING 0.3400967286036717
VR__EV_TO_EBIT_RANKING 0.28812920387373175
Intercept: 678.0111108846475
```

Coeficientes de nuestra regresión lineal para mejorar a Greenblatt

También probamos otros métodos para intentar mejorar a Greenblatt como Decision Tree, Random Forest, XGBoost, pero **decidimos quedarnos con la regresión lineal por sencillez de criterio**. Queda pendiente la comprobación de los otros coeficientes obtenidos. Asimismo decidimos, en lugar de pasar todo el dataset, usar una estrategia por ventanas de 5 y 10 años. Todo ello está en el libro de trabajo Joel Greenblatt.ipynb, en el que nos quedamos con las mismas columnas de Greenblatt.

Ahora el siguiente paso era comprobar si financieramente los coeficientes tenían sentido. Para ello hicimos un dataset de predicción en ML utilizando ventanas de tiempos.



Los resultados nos los llevamos a Tableau para realizar un backtesting financiero. Utilizamos diversos filtros financieros para seleccionar las mejores empresas y elaborar así nuestra cartera de valores. Tiene todo el sentido, por ejemplo, eliminar empresas de baja capitalización y empresas cuya volatilidad (riesgo de perder lo invertido) sea alta. Así vemos una primera aproximación a los resultados.

El estudio de Greenblatt lo continuamos eliminando outliers por año. A continuación empleamos ventanas de 5 y 10 años utilizando XGBoost. Con estos resultados metimos el resto de columnas del dataset. Siguiendo con la idea de Greenblatt, creamos rankings para todas las columnas. A continuación se mantuvo el criterio de eliminación por año de los outliers del dataset. Los resultados se llevan a la columna LR Rank que podemos ver en la Tabla Comparativa.

	S&P 500	Greenblatt	Warren Buffet	LR Rank	XGBoost Classification v1	XGBoost Classification v2
2001	-0,11	0,28	0,07	0,08	0,11	0,05
2002	-0,24	-0,02	-0,04	-0,07	0,19	0,25
2003	0,22	0,47	0,16	0,46	0,40	0,42
2004	0,09	0,43	0,04	0,18	0,20	0,29
2005	0,04	-0,05	0,01	0,62	0,11	0,32
2006	0,12	0,12	0,24	0,19	0,27	0,32
2007	0,04	0,10	0,29	0,24	0,21	0,17
2008	-0,30	-0,46	-0,32	-0,45	-0,30	-0,16
2009	0,20	0,61	0,03	0,89	1,24	1,22
2010	0,11	0,11	0,21	0,43	0,26	0,21
2011	-0,01	-0,21	-0,05	-0,20	-0,13	-0,08
2012	0,12	0,10	0,17	0,22	0,05	0,18
2013	0,26	0,22	0,33	0,40	0,46	0,19
2014	0,12	-0,07	0,27	0,14	0,05	-0,01
2015	-0,01	-0,08	-0,13	-0,01	0,20	-0,01
2016	0,11	0,01	0,23	0,09	0,25	0,16
2017	0,18	0,29	0,22	0,23	0,37	0,21
TOTAL	**0,0440**	**0,0764**	**0,0878**	**0,1605**	**0,1981**	**0,1925**

Como los resultados no eran todo lo satisfactorios que esperábamos, en lugar de una regresión lineal para predecir la rentabilidad optamos por **cambiar a un criterio de clasificación de superar o no al índice.** Se buscó un algoritmo que funcionara bien y encontramos que, para 2016, conseguíamos una precisión de 65.56% para test utilizando XGBoost. Por ello decidimos usar este algoritmo desde 2001 hasta 2016 con el método de las ventanas para poder hacer el backtesting en Tableau. (columna XGBoost Classification V1)

```
model = xgb.train(params=params, dtrain=xg_train, num_boost_round=cv.shape[0])
```

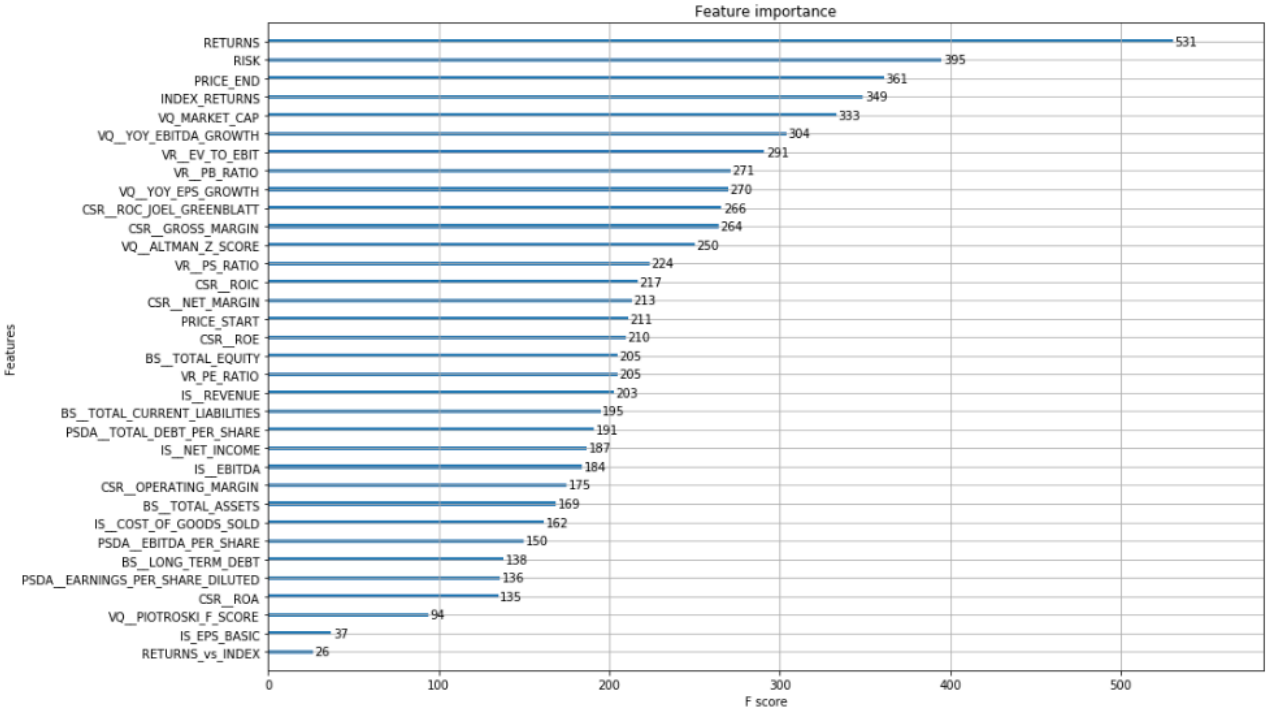
```
print("Accuracy Train: %.2f%%" % (accuracy_score(y_train,
                                                 [round(value) for value in model.predict(xg_train)]) * 100.0))
```

```
Accuracy Train: 68.77%
```

```
print("Accuracy Test: %.2f%%" % (accuracy_score(y_test,
                                                [round(value) for value in model.predict(xg_test)]) * 100.0))
```

```
Accuracy Test: 65.56%
```

Resultados de precisión en train y test del XGBoost



Importancia de las distintas variables de nuestro dataset para el target de clasificación

Ya en Tableau, volvimos a utilizar criterios financieros para filtrar empresas como haría un inversor a la hora de elegir su cartera de valores. Con ello conseguimos **batir al índice de forma consistente y continuada en el tiempo desde 2001 hasta 2017**. Ojo, no decimos que todas las empresas de nuestra cartera batan al índice o den incluso resultados positivos. Esto sería realmente imposible. Pero sí que vemos que el conjunto de nuestra cartera de valores supera al S&P 500 y además nos llevamos alguna sorpresa positiva.

En la tabla 2 vemos además la rentabilidad que habríamos obtenido de haber invertido un dólar en el año 2001.

	GANANCIAS					
	S&P 500	Greenblatt	Warren Buffet	LR Rank	XGBoost Classification v1	XGBoost Classification v2
INVERSION	1,00	1,00	1,00	1,00	1,00	1,00
2001	0,89	1,28	1,07	1,08	1,11	1,05
2002	0,68	1,25	1,02	1,01	1,33	1,32
2003	0,83	1,84	1,19	1,47	1,87	1,87
2004	0,91	2,63	1,24	1,72	2,24	2,41
2005	0,95	2,49	1,25	2,79	2,49	3,18
2006	1,06	2,80	1,55	3,32	3,16	4,22
2007	1,10	3,08	1,99	4,11	3,84	4,95
2008	0,76	1,66	1,36	2,26	2,67	4,18
2009	0,91	2,67	1,40	4,27	5,98	9,26
2010	1,01	2,97	1,69	6,08	7,53	11,21
2011	1,00	2,35	1,61	4,87	6,55	10,30
2012	1,12	2,58	1,89	5,96	6,90	12,14
2013	1,41	3,16	2,50	8,34	10,06	14,45
2014	1,59	2,93	3,18	9,48	10,56	14,27
2015	1,58	2,70	2,78	9,37	12,66	14,17
2016	1,76	2,72	3,43	10,21	15,76	16,48
2017	2,08	3,50	4,18	12,55	21,59	19,95

Tabla 2

Conclusiones

Batir a la bolsa es un problema financiero. Joel Greenblatt intentó resolver dicho problema a partir de datos fundamentales con el objetivo de ganar dinero. Su intención no era invertir en las empresas que mejor rentabilidad le fueran a dar, sino **obtener de forma consistente y repetida en el tiempo una rentabilidad**. La fórmula de Greenblatt ha funcionado en el tiempo hasta el año 2010, pero sin ofrecer un beneficio óptimo y ni siquiera ha sido su objetivo hacerlo.

Nuestro objetivo desde siempre ha sido en cierto modo similar a Greenblatt. Queríamos **determinar una cartera de valores en la que invertir y batir al índice S&P 500**. La obtención de dicha cartera desde el punto de vista de ML y DL partiendo de indicadores fundamentales se arroja realmente complicada. El comportamiento humano no está considerado en el análisis fundamental, como tampoco lo están las crisis económicas, las noticias internacionales, los cambios en materia de legislación, las declaraciones de los CEOs poco afortunadas y otros factores que vemos como diariamente impactan de forma significativa en los valores de las acciones. Con los datos que contábamos es muy difícil hacer entrenar modelos de DL y obtener buenos resultados.

Pese a la dificultad del objetivo, hemos visto que **es posible mezclar los resultados obtenidos mediante técnicas de ML y DL con el conocimiento financiero**, aplicar técnicas de visualización y análisis exploratorio en Tableau **y obtener buenos resultados** de cara a una inversión rentable a largo plazo e incluso superar al índice S&P 500 de forma consistente en los últimos años.

Se nos quedan en el tintero seguir haciendo pruebas, aplicar otros modelos, utilizar otros criterios, añadir otras fuentes de datos, incorporar información de factores externos como noticias o redes sociales…. pero superar al índice era nuestro objetivo inicial y con ello cumplimos nuestra misión.