UNIVERSIDAD GALILEO
FACULTAD DE INGENIERÍA DE SISTEMAS, INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

Reporte de Competencia de Kaggle
"House Prices"

RODRIGO ALBERTO CORDERO ALVAREZ 17001922
MARCELO ALFREDO DEL ÁGUILA MORAGA 17001380
KEVIN JOSÉ HERNÁNDEZ MARROQUÍN 17001095

INVESTIGACIÓN DE OPERACIONES I
SÉPTIMO SEMESTRE

GUATEMALA, 22 DE ARBIL DE 2020

Tabla de Contenidos

Introducción

Con la tecnología de hoy en día, ha sido posible realizar modelos que tengan la capacidad de predecir valores o situaciones en dependencia del problema que se quiere atacar.
En todo el mundo, siempre existirán problemas de predicción. Esto se hace notar considerablemente en las empresas ya que siempre buscan tomar la mejor decisión para evitar caer en situaciones no deseables (como en bancarrota). Escenarios como estos, demuestran la importancia de la capacidad de poder predecir en base a ciertos parámetros importantes.

Un problema muy común es el de poder predecir un número dado ciertos parámetros. Este se conoce como un problema de regresión.
Un problema de regresión que se hace destacar es el problema de predecir el costo de casas que están en venta en base a ciertos parámetros.

Este problema se le conoce como "House Prices" y ha sido publicado en la página de Kaggle.com como un tipo de competencia para determinar quién ha logrado hacer el mejor modelo de predicción.

Nuestra tarea como grupo, es de construir un modelo muy bueno dedicado a la predicción de los costos de casas.

Objetivos

Objetivo General:

Construir un modelo robusto que sea capaz de predecir los costos de casas (en base a ciertos parámetros) y que nos permita colocarnos en una muy buena posición en la competencia de Kaggle.

Objetivos Específicos:

- Aplicar los conocimientos adquiridos en el curso de Investigación de Operaciones I y de Seminario Profesional I.
- Realizar análisis a los Datasets presentados para obtener una idea de la naturaleza de los datos.
- Aplicar distintos modelos para alcanzar el mejor resultado.
- Aprender nuevas técnicas para el manejo de datos y aplicarlos a nuestro problema.

Marco Teórico

- **Correlación:** La Correlación en la estadística determina la relación o dependencia entre dos variables, en otras palabras, determina si los cambios de una de las variables influyen en la otra variable, de ser este el caso se dice que las variables están correlacionadas. (SuperProf, n.d.)

- **Feature Engineering:** Es un proceso que se encarga de usar "domain knowledge" de la data para crear cualidades que son necesarios para que el algoritmo del machine learning funcione, si lesas cualidades son hechas de manera correcta, incrementa la certeza de las predicciones, al crear nuevas cualidades desde raw data que facilita el proceso de aprendizaje del algoritmo. (Shekhar, n.d.)

- **Integer Encoding:** Es un proceso diseñado para tomar columnas cualitativas que no tienen sus ítems no sigan un patrón de peor a mejor, por ejemplo, sino que son cualidades separadas y convertirlas en columnas con números enteros.

  La manera en la que funciona es, se observa cuentas diferentes cualidades conforman a la columna, se crea una nueva columna por cada cualidad, y en esa columna se pone un "1" en todas las filas donde aparezca la cualidad que se está analizando dentro de la columna original y se pone un "0" en las demás filas, y así hasta tener una fila por cualidad.

- **Métricas:** en este proyecto destacaron 2 métricas que dictaron lo bueno de nuestros modelos:

  - **RMS (Root Mean Squared):** Es un algoritmo que se encarga de dar un promedio de un valor donde no se necesita o no es importante el signo de los valores, por ejemplo si se quiere obtener el promedio de los errores de un modelo , obtenernos de una manera convencional con la fórmula de la media no nos daría un valor útil debido a que los errores pueden ser negativos o positivos , ahí el RMS nos permite obtener el promedio del error del modelo, lo cual nos permite tener una métrica para calificar nuestro modelo. (M., 2019)

  - **Mean Absolute Error:** Al igual que el valor RMS el MAE nos permite obtener el promedio de las diferencias entre los valores obtenidos por el modelo y los valores reales, la diferencia entre ambas métricas es su forma de calcularla. (Urquhart, 2013)

- **Normalización:** La normalización es un proceso en el cual se toma una columna con valores enteros o reales y se restringen a valores entre 0 y 1, lo cual nos permite reducir la carga a la computadora a la hora de procesar los valores debido a que es más fácil para la computadora trabajar con ese tipo de valores.

- **One Hot Encoding:** Al igual que Integral Encoding este proceso se encarga de tomar columnas de cualitativas y convertirlas en números para poder utilizarlas en un modelo matemático, la diferencia es que este tipo de proceso se aplica a columnas cullas cualidades tengan una relación entre ellas, por ejemplo si se tiene cualidades que van de muy malo a muy bueno se le puede asignar un numero a cada cualidad por ejemplo muy malo es 0 y muy bueno es 5 lo cual nos permite mantener el mismo nivel de cualidad pero con valores numéricos.

- **R:** Es un lenguaje y un ambiente de programación dedicado a estadística computacional y gráficos, R comparte muchas similitudes con otro leguaje de programación llamado S, R es in proyecto de GNU y S es un lenguaje desarrollado por AT&T, R proporciona una amplia gama de funciones estadísticas como herramientas graficas y un ambiente Open Source. (Project, n.d.)
- **Redes Neuronales:** Es una clase de inteligencia artificial que intenta imitar la forma en la que el cerebro humano funciona, en vez de trabajar con modelos digitales , una red neuronal funciona creando conexiones entre "elementos de procesamiento", que es el equivalente computacional de una neurona, se le asigna un peso a cada conexione entre las neuronas mientas la maquina aprende lo cual le permite mejorar sus predicciones si se entrena de la manera correcta. (webopedia, 2020)

- **Regresión:** Es una clase de algoritmos en estadística que se utilizan para predecir valores o resultados en dependencia de una cantidad n de variables, las regresiones pueden ser: lineales, polinómicas, logísticas, logarítmicas, entre otras.

- **Regresión Lineal:** Una regresión lineal es un algoritmo de predicción que se encarda de generar una recta de predicción que se acople de la mejor manera a los valores que se tengan, una de sus características mas importantes es que la distancia entre los valores reales a la recta de predicción sumados da 0, eso nos permite obtener una sola recta de interés.

- **Tensorflow:** Es una librería open source y compatible con Python que facilita el desarrollo de Machine learning. (Yegulalp, 2019)

- **Loss:** En el entrenamiento de una red neuronal se requiere que para el modelo se escoja una función loss, en base las funciones loss o funciones objetivo es la función que se encarga de evaluar las posibles soluciones.

- **Regresión Ridge:** Conocido como regresión contraída, este tipo de regresión impone una penalización al tamaño de los coeficientes. Los coeficientes minimizan la suma de los cuadrados ya que agrega el cuadrado de la norma del vector que formaron los coeficientes, utilizando un parámetro λ con el cual se controla que tanto es la penalización.

- **Regularización Lasso:** Muy parecido a la Regresión Ridge ya que añade una penalización a los coeficientes, pero con el detalle que al encontrar variables que no aportan significativamente, al modelo las elimina.

- **Regularización ElasticNet:** Con esta regularización se hace una combinación de la Regularización Lasso y la Regulación Lasso la cual se logra añadiendo un parámetro *r,* este parámetro funciona para darle una importancia relativa a cada una de las regulaciones.

- **Arboles de decisión:** Es un mapa con el que se obtiene un distinto resultado debido a la serie de decisiones relacionados. Los árboles de decisión comienzan con un nodo individual y luego se dividen en distintos nodos adicionales, lo que le da una forma similar a un árbol. Este método tiene el problema de hacer *overfitting.*

- **Pureza de Gini:** Es una forma de medir la similitud de dos datos seleccionados al azar de una misma población ya que si ambos datos demuestran las mismas características la pureza es de 1.

- **Bagging:** Se creó al observar los problemas que causan los árboles al sobre entrenarse. Este método crea muchos árboles en paralelo lo que forma un bosque. Cada árbol aporta con su predicción y se toma la media de estas en el caso de variables continuas y la clase mayor frecuente en las variables discretas.

- **Bosque Aleatorio:** Para trabajar utiliza un conjunto de árboles de decisión, los cuales forma de manera automática. Utiliza bagging, lo cual hace que los árboles solo vean una pequeña fracción de datos, provocando que cada árbol se entrene con distintas variables. Formar distintos árboles de esta manera logra compensar errores de los árboles anteriores logrando una predicción mucho mejor.

- **Boosting:** Fue creado para problemas de clasificación, aunque luego fue extendido para ser utilizado para problemas de regresión. Es una idea muy poderosa creada para el año de 1990 ya que combina clasificadores "débiles" y de esta forma disminuye las características de cada uno para lograr un clasificador más eficiente.

- **Boosting Adaptivo (AdaBoost):** Es un algoritmo que utiliza un entrenamiento iterativo de los clasificadores débiles dándole mayor importancia a los datos que se han clasificado mal y forman un nuevo clasificador. Se le da el nombre de adaptivo debido a su forma de adaptarse a los datos mal predichos y de esta forma mejorar su precisión.

- **Potenciador del Gradiente:** Encontrado habitualmente como Gradient Boost es una de las técnicas más poderosas hasta este año para construir modelos de predicción. A diferencia del boosting adaptivo que utiliza el peso de cada variable, el potenciador del gradiente utiliza la conocida "loss function", una forma de medir que tanto se aleja la predicción del valor correcto. Entrena muchos modelos de una forma gradual y los suma de una forma secuencial.

Problema

Situación por Enfrentarse:

El problema que nos enfrentamos, según la página oficial de Kaggle, es la predicción del valor de una casa en base a ciertos parámetros. Quieren demostrar que el precio de una casa depende más que solo el material con el que está hecho, el número de cuartos, la ciudad en dónde se encuentra, etc.

Nosotros podemos deducir que este problema es de tipo de regresión ya que se debe predecir un valor continuo y no un valor categórico.

En la página oficial de Kaggle se menciona una breve introducción al problema y nos proporciona ciertos archivos que nos ayudarán a resolver este problema.

Entre los archivos que nos proporciona podemos encontrar:

- *train.csv*: Archivo que contiene el dataset para el entrenamiento del modelo.
- *test.csv*: Archivo que contiene el dataset para que el modelo prediga los precios de las casas.
- *data_description.txt*: Archivo que contiene una descripción detallada de todas las variables que se encuentran en los datasets.
- *sample_submission.csv*: Archivo que muestra un modelo de cómo se debe subir las predicciones a la competencia.

La descripción que nos muestra Kaggle sobre los dataset es que hay un total de 79 variables y en base a ellas se quiere predecir 'SalePrice', que es el costo de la casa; este es el objetivo del modelo a construir.

Problemas Potenciales

- Poco conocimiento y poca práctica en el área de Ciencias de Datos.
- Muchas variables explicativas; hay tantas variables categóricas como cuantitativas.
- Disponemos de 1460 datos para entrenamiento, consideramos que es poco.
- No conocemos demasiadas herramientas para armar un modelo robusto.
- Es una competencia abierta para participantes de todos los niveles, lo que dificulta alcanzar un puesto alto.

Approaches

Los modelos utilizados son los siguientes:

- Regresiones Lineales
- Redes Neuronales
- Regresiones Lineales con XGBoost

Regresiones Lineales

Las regresiones lineales pueden ser el tipo de regresiones más sencillo y de los menos exactos en todos los casos, pero nosotros decidimos utilizar este modelo de regresiones debidos a que teníamos una cantidad muy alta de variables que nos permitiría hacer una hiper recta de +80 dimensiones para poder obtener por medio de la función de summary() las variables más importantes o con más peso.

Después de limpiar toda la data, darle un formato, normalizándola y aplicándoles One Hot Encoding e Integral Encoding, generamos una regresión lineal de +80 variables y fuimos descartando de a poco las variables con menos peso hasta que obtuvimos 33 variables con tres estrellas, y nos decidimos a hacer 3 modelos de 11 variables y observar cuales variables nos daban el mejor resultado y unir los 3 modelos y obtener nuestro modelo final.

Luego de obtener los modelos logramos tener un modelo que nos daba el mejor resultado posible con ese modelo, que no era malo, pero no era el mejor.

```
lm(formula = SalePrice ~ BsmtExposure + LotFrontage + WoodDeckSF +
    X2ndFlrSF + OpenPorchSF + BsmtFinType1 + GarageQual + LotShape_Reg +
    LotShape_IR2 + LotShape_IR3 + LotArea, data = dataTrain)

Residuals:
    Min      1Q  Median      3Q     Max
-322332  -34395   -6073   23329  372069

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    3911.2     8522.4   0.459  0.64635
BsmtExposure  16663.7     1604.4  10.386  < 2e-16 ***
LotFrontage  212462.9    23269.3   9.131  < 2e-16 ***
WoodDeckSF    78102.6    11021.9   7.086 2.15e-12 ***
X2ndFlrSF    106303.7     7550.6  14.079  < 2e-16 ***
OpenPorchSF  108712.1    13029.9   8.343  < 2e-16 ***
BsmtFinType1   6312.3      785.6   8.035 1.93e-15 ***
GarageQual    16873.0     2151.0   7.844 8.40e-15 ***
LotShape_Reg -14268.4     3376.7  -4.225 2.53e-05 ***
LotShape_IR2   8076.6     9665.7   0.836  0.40352
LotShape_IR3 -51701.2    19269.6  -2.683  0.00738 **
LotArea      121529.5    37258.6   3.262  0.00113 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 57980 on 1448 degrees of freedom
Multiple R-squared:  0.4714,    Adjusted R-squared:  0.4674
F-statistic: 117.4 on 11 and 1448 DF,  p-value: < 2.2e-16
```

Como podemos observar en la imagen anterior el modelo no solo consta de variables de 3 estrellas y eso se debe a que un modelo solo con 3 estrellas no nos daba un r^2 mayor a 0.30 lo cual es muy bajo, pero con el modelo actual a pesar de solo tener un $r^2 = 0.4674$ nos permitió tener un valor RMSE de 0.338 el cual no es malo, pero para los estándares de la competencia no era suficiente.

Regresiones Lineales con XGBoost

Extra Gradiente Boosting utiliza árboles de decisión para hacer la predicción sobre una variable a través del aprendizaje automático realizado a un conjunto de datos.

Al utilizar clasificadores débiles, en este caso árboles, pero potenciando los resultados que cada uno provee, puede generar una predicción correcta o bastante cerca del valor correcto. Su forma de potenciar dichos resultados es procesar de forma secuencial los datos con una función de pérdida, lo cual minimiza el error de cada iteración con respecto a la anterior.

Una de las mayores ventajas de utilizar este modelo es que puede manejar datos faltantes.

Sus desventajas es la cantidad de RAM que demanda ya que el algoritmo procesa por medio de validación cruzada, es decir, conforme va iterando guarda las variables que mayor aportación le van creando al modelo. Otra desventaja es la cantidad de parámetros que maneja, necesitando ajustar correctamente cada una para minimizar el error de precisión.

Los parámetros modificados para la construcción del modelo fueron:

- booster, en este casi árboles de decisión.
- rounds, el número de iteraciones.
- Max Depth, la cantidad de sub-nodos permitidos a crear.
- Eta, es su velocidad de aprendizaje, a menor eta se aproxima de una forma más cuidadosa al valor correcto, pero toma más tiempo.
- Lambda, un parámetro de regularización utilizado para que el valor de ganancia de cada nodo disminuya, haciéndolo menos sensible a los datos, evitando que haya sobre aprendizaje.
- Gamma, utilizado para eliminar los nodos que no aportan al modelo.
- Verbose, permite ver cada iteración del modelo.
- Seed, al formar los árboles de forma aleatoria, se selecciona un número para asegurar que con cada prueba se elijan los mismos árboles y se puedan comparar los resultados.

La forma en la que se evalúa este modelo es con el error de mínimos cuadrados (rmse).

Redes Neuronales

Las redes neuronales pueden ser modelos muy eficaces tanto para los problemas de regresión como los problemas de clasificación.
Se decidió utilizar redes neuronales debido a que un integrante del grupo tiene experiencia moderada con dichos modelos.

El reto de las redes neuronales es encontrar la mejor arquitectura que pueda realizar muy buenas predicciones.

El approach que se tomó es armar una arquitectura cuyas capas ocultas están compuestas por una cantidad de neuronas en potencias de 2 y decrecientes e ir agregando o quitando en dependencia de los resultados que proporcione la página de Kaggle.

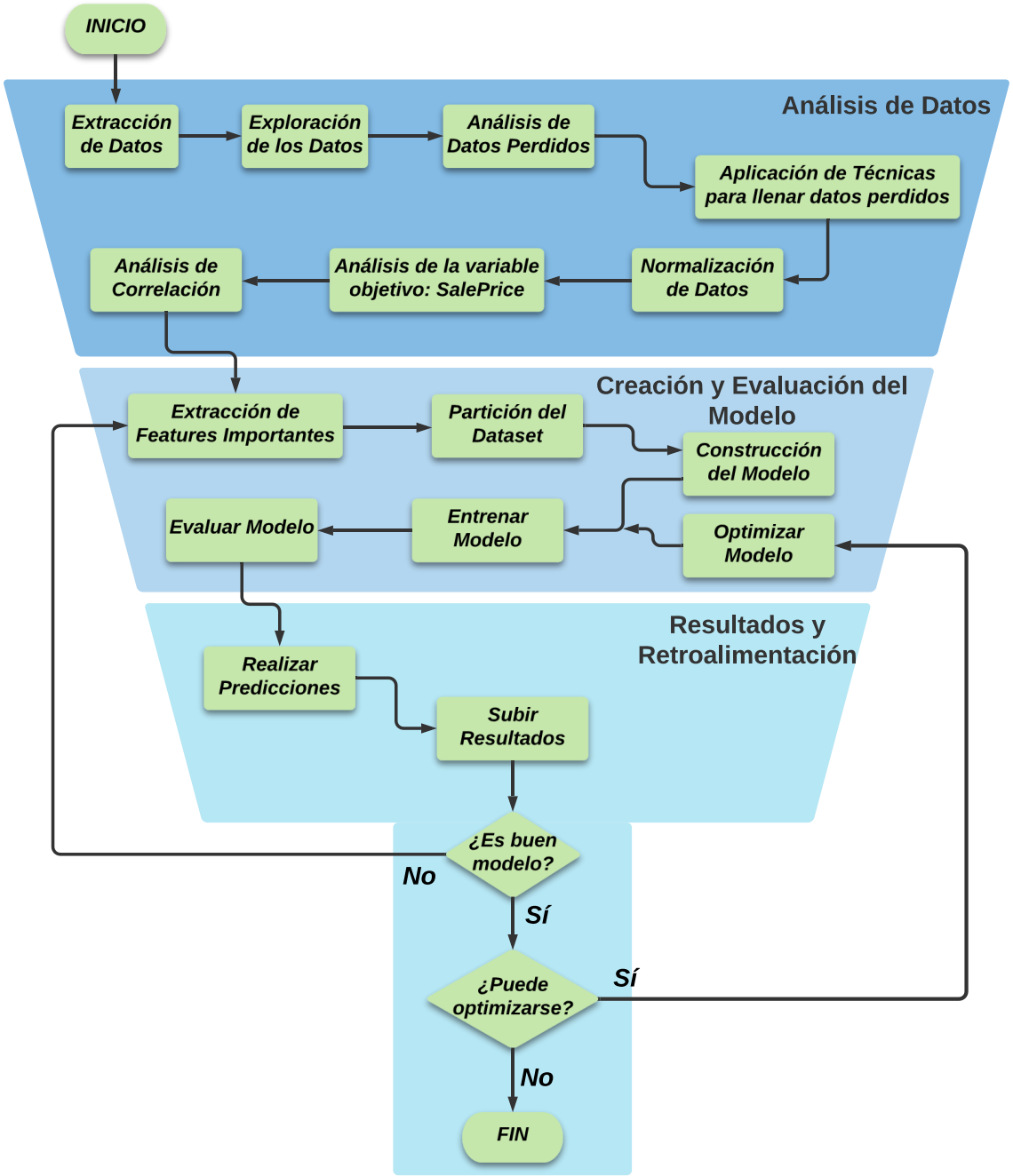
Para las redes neuronales, hay una cantidad muy grande de parámetros que se pueden modificar, sin embargo, para reducir dicha cantidad, solo se enfocará en modificar:

- El número de capas ocultas de la arquitectura.
- El número de neuronas por capa oculta.
- Inicialización de pesos de las neuronas.
- El optimizador de la red.
- La métrica de la red.
- Batch Size y Validation Split.
- Número de Epochs.

Para evaluar qué tan buena es la red, se utiliza la métrica de Loss.

Metodología

La metodología utilizada para la realización de este proyecto es el siguiente:



Detalles de la Metodología

Fase de Análisis de Datos

1. Extracción de Datos:
   Consiste en extraer los datos contenidos en los archivos *train.csv* y *test.csv* y construir los datasets de train y test, respectivamente.

2. Exploración de los Datos:
   Consiste en conocer ambos datasets (train y test) y observar qué features tiene cada uno, el tipo de dato de cada feature (si es categórico o cuantitativo) y cuántos datos tiene cada dataset. Se decidió combinar ambos datasets en uno solo para analizar los datos.
   En este paso se utilizó mucho el archivo *data_description.txt* para ir analizando qué significa cada feature y los posibles valores que podía tener.

3. Análisis de Datos Perdidos:
   Consiste en conocer qué features tienen "Missing Values", valores Nulos o valores de tipo NaN. Se consultó muy frecuentemente el archivo *data_description.txt*, ya que especificaba que algunos features que poseen Missing Values o NaN sí representaban un valor como tal.

   Después de haber hecho el análisis descrito anteriormente, se procede a aplicar técnicas[1] para el reemplazo y rellenado de Missing Values o NaNs.

4. Encoding:
   Consiste en aplicar las técnicas de Integer Encoding o One Hot Encoding a cada feature de tipo categórico, según sea necesario.

5. Normalización de Datos:
   Consiste en normalizar los datos de cada feature para disminuir el tiempo de entrenamiento del modelo y mejorar su capacidad para predecir.

6. Análisis de la variable objetivo '*SalePrice*':
   Consiste en realizar un análisis estadístico a la variable SalePrice:
   - Analizar su distribución
   - El sesgo de la gráfica de distribución
   - Media
   - Varianza

7. Análisis de Correlación:
   Consiste en analizar la correlación de todos los features con la variable objetivo: SalePrice.
   Además, se realiza un mapa de calor de correlación para facilitar la visualización de las correlaciones entre los distintos features.

8. Separación del Dataset:
   Consiste en separar el dataset combinado (en el dataset de train y test) después de haber terminado el análisis de datos.

Fase de Creación y Evaluación del Modelo

1. Extracción de Features Importantes:
   En base al análisis de correlación, se extraen los features que cumplen con el criterio de "Threshold", es decir, si la correlación de un determinado feature con SalePrice es igual o mayor a un Threshold establecido, entonces es considerado como un feature importante.

   Por último, se crea un nuevo dataset '*train*' compuesto únicamente por el conjunto de features extraídos anteriormente.

2. Partición del Dataset:
   Consiste en hacer un shuffle al nuevo dataset y particionarlo en 2 nuevos datasets: *newTrain* y *evaluation*. El propósito del dataset *newTrain* es ser utilizado para el entrenamiento del modelo, y el propósito del dataset *evaluation* es para evaluar qué tan bueno es nuestro modelo. Se utiliza un criterio de 80% de los datos para newTrain y 20% para *evaluation* y

3. Construcción del Modelo:
   Consiste en crear el modelo ya sea usando una regresión lineal, regresión XGBoost o una red neuronal.

4. Entrenar Modelo:
   Consiste en utilizar el dataset *newTrain* para entrenar el modelo construido anteriormente.

5. Evaluar Modelo:
   Consiste en utilizar el dataset *evaluation* para evaluar el desempeño del modelo.
   Para los modelos de regresión lineal o regresión XGBoost se utilizó la métrica RMSE para evaluar el desempeño.
   Para los modelos de tipo red neuronal, se utilizó la métrica Mae y Loss, y en algunos casos RMS y Loss.

Fase de Resultados y Retroalimentación

1. Realizar Predicciones:
   Consiste en utilizar el modelo para predecir la variable objetivo (SalePrice) para los distintos datos contenidos en el dataset *test*.

2. Subir Resultados:
   Consiste en crear el archivo con el formato especificado en Kaggle (usando las predicciones obtenidas anteriormente) para luego subir nuestras predicciones y así obtener un score y avanzar en la tabla de Leaderboard.

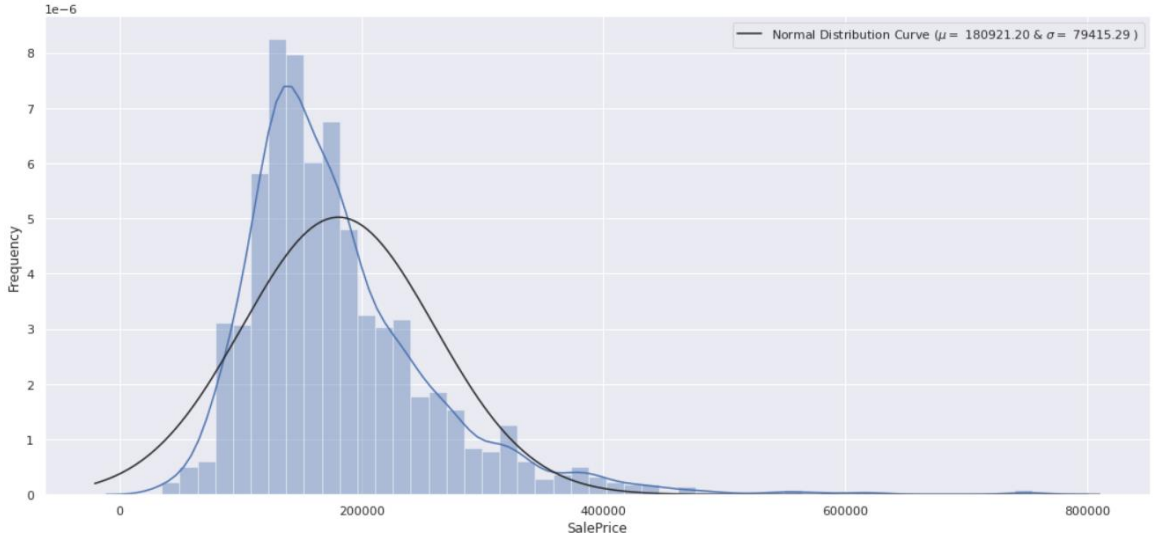
3. (Paso Extra) Optimizar Modelo:
   Luego de obtener el respectivo punteo en Kaggle, se decide si vale la pena optimizar el modelo cambiando ciertos parámetros según sea el modelo.

Hallazgos

Fase de Análisis de Datos

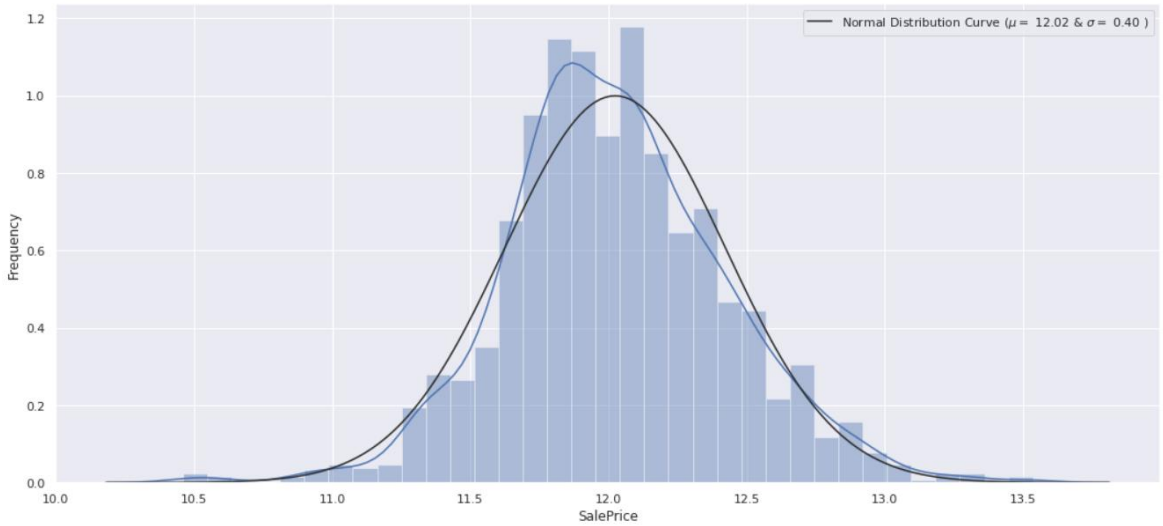
- Análisis de Datos Perdidos:
  Se encontraron varios features que poseen Missing Values o NaNs. Sin embargo, el archivo no describía que dichos features tenían la posibilidad tener un valor NaN o un "Missing Value".
  Para las técnicas de reemplazo y rellenado de Missing Values o Nans, se decidió que:
  - Para el feature *'LotFrontage',* los valores NaN son reemplazados por la mediana del feature.
  - Para los Features *'GarageyrBlt'*, *'GarageArea'*, *'GarageCars'*, *'BsmtFinSF1'*, *'BsmtFinSF2'*, *'BsmtUnfSF'*, *'TotalBsmSF'*, *'BsmtFullBath'*, *'BsmtHalfBath'*, *'BsmtFinSF1'* y *'MasVnrArea'*, los valores NaN son reemplazados por un 0.
  - Para el feature *'MSZoning'*, los valores NaN son reemplazados por la moda del feature.

- Análisis de la variable objetivo '*SalePrice*':
  Se descubre que la gráfica de distribución de frecuencias de la variable SalePrice se encuentra sesgada hacia la izquierda, además se descubre que dicha gráfica tiene una curtosis de tipo Leptocúrtica.



Gráfica de Distribución de Frecuencias de SalePrice

A la variable objetivo se le aplica una transformación de tipo Log(p+1) con el fin de normalizar la data.



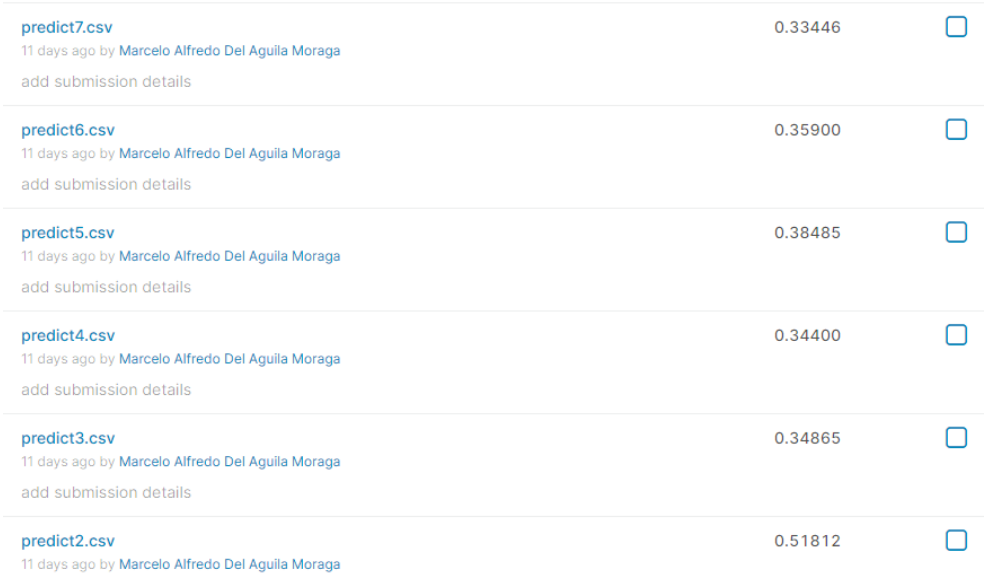
Gráfica de Distribución de Frecuencias de SalePrice luego al aplicarle la transformación Log(p+1)

Resultados

Resultados regresiones lineales y polinómicas

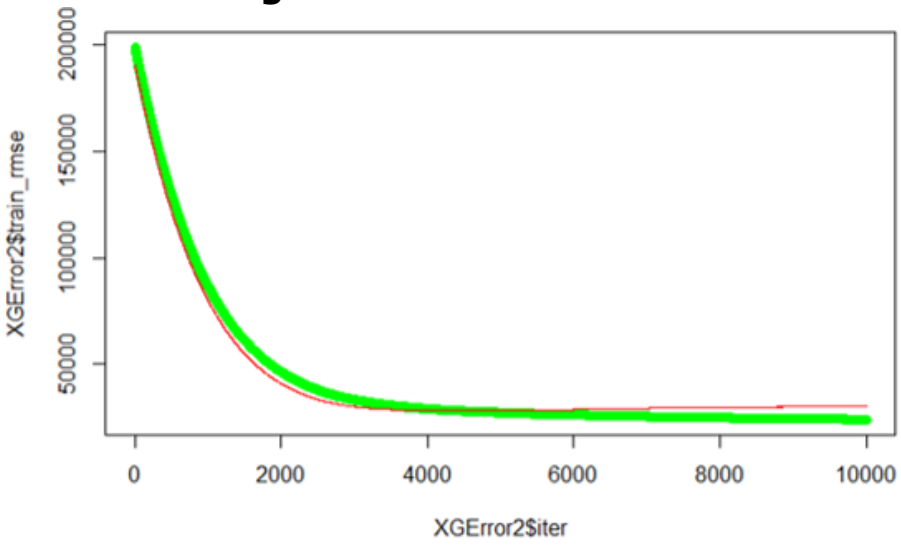
De los modelos realizados con regresiones lineales nuestros modelos fueron variando mucho, pero ninguno nos llevó a la meta esperada.

De parte de Marcelo del Aguila, los mejores resultados obtenidos con una regresión lineal de 15 variables fueron los siguientes:



A pesar de la mejora no nos llevó más lejos que el P_{30}.

Resultados regresiones XGBoost



Resultados redes neuronales

HP_v2_tensorflow (9).csv 2 days ago by Kevin Hernandez Try # 10	0.14164	☐
HP_v2_tensorflow (8).csv 4 days ago by Kevin Hernandez Try #9	0.13637	☐
HP_v2_tensorflow (7).csv 4 days ago by Kevin Hernandez Try #8	0.17374	☐

Resultados del mejor modelo que realizamos

XGkaggleOut.csv 2 days ago by Rodrigo Cordero add submission details	0.12657
XGkaggleOut.csv 2 days ago by Rodrigo Cordero add submission details	0.12657
XGkaggleOut.csv 2 days ago by Rodrigo Cordero add submission details	0.12585
XGkaggleOut.csv 2 days ago by Rodrigo Cordero add submission details	0.12675
XGkaggleOut.csv 2 days ago by Rodrigo Cordero add submission details	0.12876

Conclusiones

- El paso fundamental es hacer un análisis del dataset previo a construir los modelos.
- Es muy importante realizar un análisis a todas las variables involucradas para obtener una idea de la naturaleza de los datos.
- Es necesario aplicar técnicas de 'Feature Engineering' y de limpieza de datos para mejorar el desempeño de los modelos a construir.
- Es difícil encontrar cómo 'mejorar' un modelo y asegurar que esa 'mejora' sea algo positivo y no sea perjudicial.
- El desempeño de las redes neuronales decrece conforme se aumenta la complejidad de este.

Observaciones

- Al principio nos dispusimos en resolver el problema con únicamente regresiones lineales, pero debido a que las mismas no nos llevaron muy lejos en el ranking, entonces decidimos que uno de nosotros se fuera por otro camino, con Redes neuronales el cual fue más productivo

- Cuando se empezó a limpiar la data y a seleccionar las variables que más efecto tenían en el modelo, hubiera sido mejor haber tomado grupos de 10 en 10 e ir obteniéndolas variables más impactantes de esa manera porque al obtener un modelo con 80 variables la función summary pierde fiabilidad.

Recomendaciones

- Es recomendable siempre limpiar su data antes de analizarlo y leer las instrucciones para ver si algunos NaN o Null tienen significado útil.

- Se recomienda el uso de Redes neuronales para este problema, debido a al alto número de variables que algunas tienen correlación entre ellas, y el entrenar bien las redes neuronales provee un modelo muy bueno.

- Se recomienda el uso de Hot One Encoding e Integral Encoding debido a que hay muchas columnas que son cualitativas que llegan a ser muy importantes para los modelos, ya sea Regresión o Red neuronal.

- Se recomienda el uso de TensorFlow, ya que el declarar y entrenar la red neuronal en esta herramienta de Python es mucho más amigable que en R.

- Se recomienda el uso de la función summary() de R debido a que esta función nos permite saber que variables realmente hacen una diferencia y cuales podrían ser obviadas.

Anexos

- **Repositorio de GitHub:**
  https://github.com/corderobot/Kaggle_HousePrices

Referencias Bibliográficas

M., M. (24 de Octube de 2019). *Media cuadrática*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Media_cuadrática#Media_cuadrática_de_la_velocidad_de_un_gas

Project, R. (s.f.). *What is R?* Obtenido de r-project: https://www.r-project.org/about.html

Shekhar, A. (s.f.). *LetsLearnAI: What Is Feature Engineering for Machine Learning?* Obtenido de Medium: https://medium.com/mindorks/what-is-feature-engineering-for-machine-learning-d8ba3158d97a

SuperProf. (s.f.). *Qué significa correlación estadística en Matemáticas*. Obtenido de SuperProf Diccionario: https://www.superprof.es/diccionario/matematicas/estadistica/correlacion-estadistica.html

Urquhart, B. (2013). *Mean Absolute Error*. Obtenido de ScienceDirect: https://www.sciencedirect.com/topics/engineering/mean-absolute-error

webopedia. (2020). *neural network*. Obtenido de webopedia: https://www.webopedia.com/TERM/N/neural_network.html

Yegulalp, S. (18 de Junio de 2019). *What is TensorFlow? The machine learning library explained*. Obtenido de Info World: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html