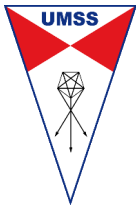
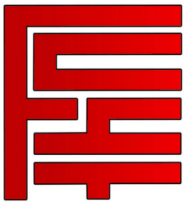


UNIVERSIDAD MAYOR DE SAN SIMÓN
FACULTAD DE CIENCIAS Y TECNOLOGÍA



Práctica Primer Parcial

Modelos Regresionales y de decisión
(Dataset: Car Sales)

Estudiantes: Aguilar Villalobos Fernanda
Fernández Sánchez Reilly Daria
Guzmán Galarza Violeta Belén
Maguiña Apaza Pablo Saul
Menacho Flores Dennys Lee

Docente: Rodriguez Bilbao Patricia.

Materia: Ciencia de Datos y Machine Learning

28/10/2022

Cochabamba - Bolivia

Índice

1. Introducción

En el siguiente trabajo se presentará el proceso de aplicar el aprendizaje automático a través de modelos regresionales y de decisión para predecir:

a) El precio de un automóvil, mediante un modelo regresional según parámetros determinados. En el mercado de la venta de autos puede haber una gran variabilidad de precios de acuerdo a varios factores como la marca, el modelo, el estado, si es nuevo o usado, etc. Por lo cual por medio de la predicción de precios se obtendrá un precio estimado para tomar en cuenta al momento de realizar una compra y comparar con los precios ofrecidos.

b) El año del modelo de un automóvil, mediante un modelo de decisión según parámetros determinados.

2. Objetivos

Realizar la evaluación e implementación de procesos de Aprendizaje Automático , a través del análisis del dataset de venta de autos para predecir tanto el precio de un auto con un modelo regresional, como el año del modelo de un auto con un modelo de decisión.

3. Marco Teórico

3.1. Venta de automóviles

El mercado de la venta de automóviles suele crecer con el paso del tiempo debido a la necesidad de las personas en comprar un automóvil por diversos factores como la capacidad de poder movilizarse de un lugar a otro, la sensación de libertad, uso como herramienta de trabajo, etc.

3.2. Machine Learning

El aprendizaje automático es una aplicación de IA que permite que los sistemas aprendan y mejoren a partir de la experiencia sin ser programados explícitamente. El aprendizaje automático se centra en el desarrollo de programas informáticos que pueden acceder a los datos y utilizarlos para aprender por sí mismos.

3.3. Aprendizaje Automático

3.3.1. Aprendizaje Supervisado

Los algoritmos de aprendizaje supervisado basan su aprendizaje en un juego de datos de entrenamiento previamente etiquetados. Por etiquetado entendemos que para cada ocurrencia del juego de datos de entrenamiento

conocemos el valor de su atributo objetivo. Esto le permitirá al algoritmo poder "aprender" una función capaz de predecir el atributo objetivo para un juego de datos nuevo.

3.3.2. Aprendizaje no supervisado

Los métodos no supervisados (unsupervised methods) son algoritmos que basan su proceso de entrenamiento en un juego de datos sin etiquetas o clases previamente definidas. Es decir, a priori no se conoce ningún valor objetivo o de clase, ya sea categórico o numérico. El aprendizaje no supervisado está dedicado a las tareas de agrupamiento, también llamadas clustering o segmentación, donde su objetivo es encontrar grupos similares en el conjunto de datos.

3.4. Regresión Lineal

Una regresión lineal se utiliza para generar predicciones sobre una variable que llamamos la variable dependiente, generalmente denominada con una y, dadas una o varias variables independientes generalmente denominadas las x. Cuando usamos regresión lineal, la variable dependiente 'y' siempre es numérica. Por ejemplo, precios de las casas, estaturas, la distancia de los planetas al sol, etc. Una regresión lineal en su forma más simple (cuando contamos con solo una variable independiente x) es una línea recta en dos dimensiones que mejor se ajusta a los valores de los datos.

$$Y_i = (a + bX_i) + \epsilon_i$$

3.5. Regresión Polinomial

La regresión polinomial es un algoritmo de regresión que modela la relación entre una variable dependiente (y) e independiente (x) como un polinomio de grado n. La ecuación de regresión polinomial se da a continuación:

$$y = a + bx + cx^2 + dx^3 \ldots$$

3.6. Random Forest

Random Forest es un algoritmo de aprendizaje automático que pertenece a la técnica de aprendizaje supervisado. Se puede usar para problemas de clasificación y regresión en ML. Se basa en el concepto de aprendizaje conjunto, que es un

proceso de combinación de múltiples clasificadores para resolver un problema complejo y mejorar el rendimiento del modelo.

Random Forest es un clasificador que contiene una serie de árboles de decisión en varios subconjuntos del conjunto de datos dado y toma el promedio para mejorar la precisión predictiva de ese conjunto de datos. En lugar de depender de un árbol de decisión, el bosque aleatorio toma la predicción de cada árbol y, en función de los votos mayoritarios de las predicciones, predice el resultado final.

El mayor número de árboles en el bosque conduce a una mayor precisión y evita el problema del sobreajuste.

3.7. Decision Tree Regression

Decision Tree es una herramienta de toma de decisiones que utiliza una estructura de árbol similar a un diagrama de flujo o es un modelo de decisiones y todos sus posibles resultados, incluidos los resultados, los costos de entrada y la utilidad.
El algoritmo del árbol de decisiones cae dentro de la categoría de algoritmos de aprendizaje supervisado. Funciona tanto para variables de salida continuas como categóricas.

3.8. Regresión Logística

La regresión logística es un modelo estadístico para estudiar las relaciones entre un conjunto de variables cualitativas Xi y una variable cualitativa Y. Se trata de un modelo lineal generalizado que utiliza una función logística como función de enlace.

3.9. SVM

El objetivo del algoritmo SVM es crear la mejor línea o límite de decisión que pueda segregar el espacio n-dimensional en clases para que podamos colocar fácilmente el nuevo punto de datos en la categoría correcta en el futuro. Este límite de mejor decisión se llama hiperplano. SVM elige los puntos/vectores extremos que ayudan a crear el hiperplano.

3.10. K-neighbors

Es un algoritmo de Machine Learning que puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Sirve esencialmente para clasificar valores buscando los puntos de datos “más similares” (por cercanía) aprendidos en la etapa de entrenamiento y haciendo conjeturas de nuevos puntos basado en esa clasificación.

3.11. **Modelo a implementar** Se aplicarán 2 modelos uno regresional con 5 algoritmos diferentes y otro modelo de clasificación con 5 algoritmos diferentes, esto de acuerdo a lo solicitado en el planteamiento original del problema, los algoritmos y datos a implementar están seleccionados de acuerdo al análisis exploratorio de la información y la correlación de la misma contra las variables objetivo.

4. Ingeniería

4.1. Descripción, ampliación y/o abstracción del problema

El problema consta de escenarios:

1.- Escenario a resolver mediante un modelo de regresión para la predicción del precio de venta de un automóvil.

2.- Escenario a resolver mediante un modelo de clasificación para la predicción del año de un vehículo asociado a un registro de venta.

4.2. Referencias de tecnologías a utilizar

Las tecnologías a utilizar para la resolución del problema son:

- Google Collaboratory, como entorno de desarrollo.
- Python, como lenguaje de programación.
- Librerías Python, para el manejo de algoritmos de Machine Learning.

  sklearn import metrics sklearn.metrics import confusion_matrix sklearn.metrics import r2_score sklearn.metrics import f1_score sklearn.metrics import median_absolute_error sklearn.metrics import explained_variance_score sklearn.metrics import max_error sklearn.metrics import balanced_accuracy_score, sklearn modelos de regresion y sklearn modelos de clasificación

- Librerías Python, para el manejo de data frames.
  numpy panda
- Librerías Python, para el manejo de gráficos.
  matplotlib.pyplot seaborn
-

4.3. Actividades y resultados

a) Pre-Procesamiento de datos

a.1) Importamos y cargamos el archivo CSV

a.2)Exploración y entendimiento de los datos

A continuación se presentan ejemplos de procedimientos sobre columnas de datos que fueron tratadas:

Descarte: Existen columnas completas con valores faltantes o nulos por lo que no serán necesarias para el análisis y podrán ser eliminadas en su totalidad:

```
[ ]  #Quitamos columnas completamente nulas por no tener datos relavantes y  evitar explosion de dimensiones
     df.drop(['latitude','longitude','mapAddress','VIN'], axis=1, inplace = True)  #Columnas enteramente vacias
```

Adicionalmente se identifican columnas con datos muy variables como el identificador de venta (AdID) y el (title) con texto descriptivo que no ayudarán al análisis de problema, por lo que dichas columnas serán eliminadas

```
[ ]  #Quitamos columnas con datos no relevantes para el analisis
     df.drop(['title','AdID'], axis=1, inplace = True)  #Columnas no tiene relevancia para los calculos
```

Se buscan y eliminan filas duplicadas:

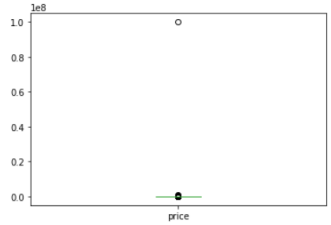
```
#Se eliminan filas duplicadas
print(f'Tamaño del set antes de eliminar las filas repetidas: {df.shape}')
df.drop_duplicates(inplace=True)
print(f'Tamaño del set después de eliminar las filas repetidas: {df.shape}')
```

```
Tamaño del set antes de eliminar las filas repetidas: (51343, 22)
Tamaño del set después de eliminar las filas repetidas: (51292, 22)
```

En la columna precio se observaron muchos muchos valores extremos atípicos, adicionalmente se ven anomalías en kilometros, valores extremos, columnas year y visit tambien con valores extremos

```
df.price.plot.box() #GRafico de caja de la columna precio
```

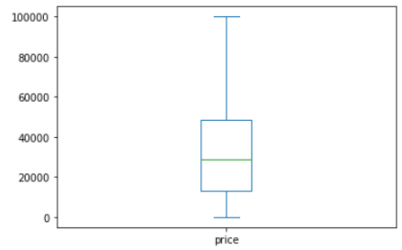
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef51321e50>
```



Borramos todos los valores superiores a 100.000 USD por estar considerados atípicos y constituir menos del 10% del total

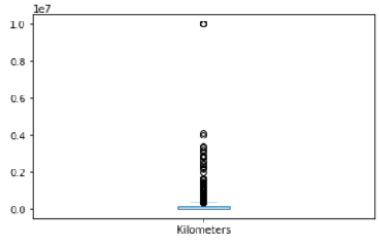
```
df.price.plot.box() # Validamos nueva distribucion de caja sin valores arriba de 100,000 USD
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef51280750>
```



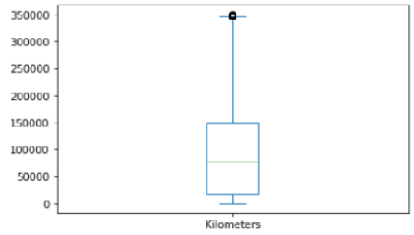
Para Kilometers eliminaremos valores atípicos por encima de los 350.000 Km, esto debido a que no constituyen un porcentaje importante y los mismos se encuentran por fuera del análisis de percentiles:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef51184090>
```




```
df.Kilometers.plot.box()
```

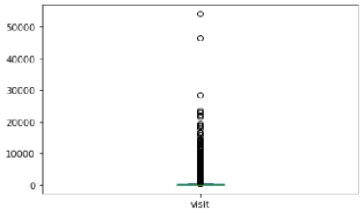
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef52701510>
```



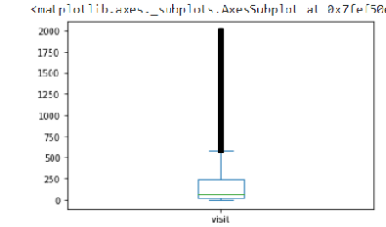
El campo visit después de la técnica de imputación de datos simple se eliminaran los valores atípicos:

```
df.visit.plot.box()
```

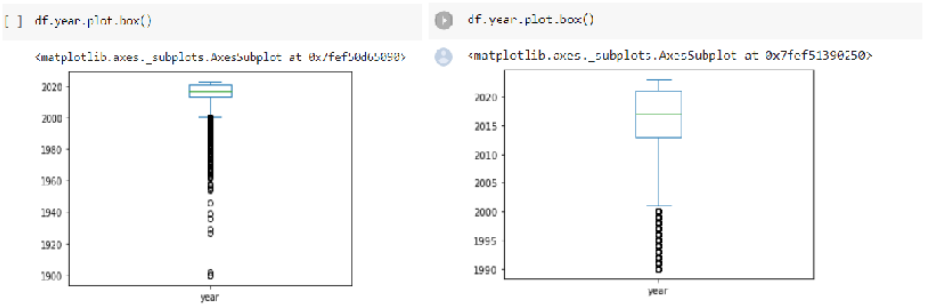
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef50f74590>
```




```
[ ] df.visit.plot.box()
```

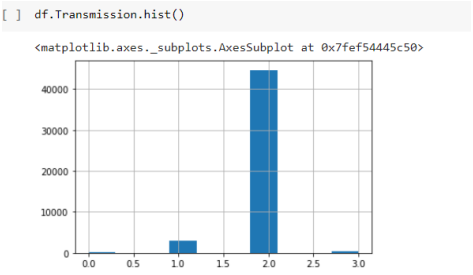


Para year también se deben eliminar valore atípicos que serán eliminados:

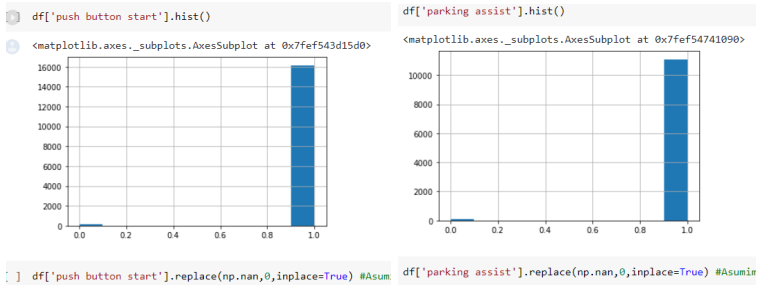


Técnica de imputación de datos faltantes - Simple: Para visit se remplazan los valores nulos con la media, esto debido a la poca cantidad de los mismos

Para Transmission también se toma el valor más común, que es 2:



lógica: Para algunos campos se llegó a la conclusión lógica que solo los que tenían habían llenado ese campo, por ende no se podría aplicar la media, en vez se llenó con 0: push button start, parking assist, sunroof, alloy wheels, debido a que este es el valor más probable según la lógica de ventas, no tanto así alguna métrica estadística .



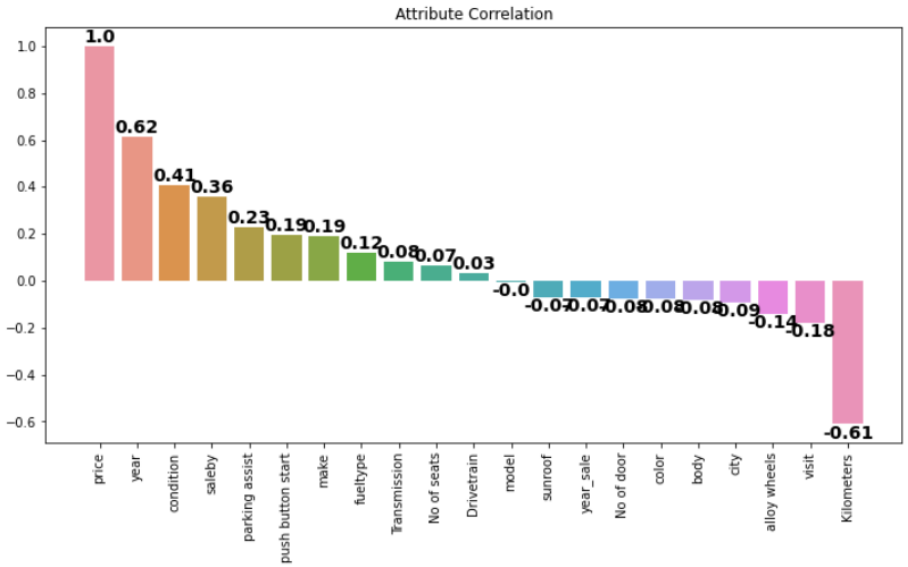
Se convirtieron las columnas categóricas a numéricas y se concluyó el depurado e imputación de todas las columnas, quedando sólo las

relevantes para el análisis y tratamiento del problema con los datos normalizados.

b) Determinación de datos entrenamiento del modelo

Modelo de Regresión:
Se determinó que para el modelo de regresión se predecirá el precio del auto (atributo target **y = price**).

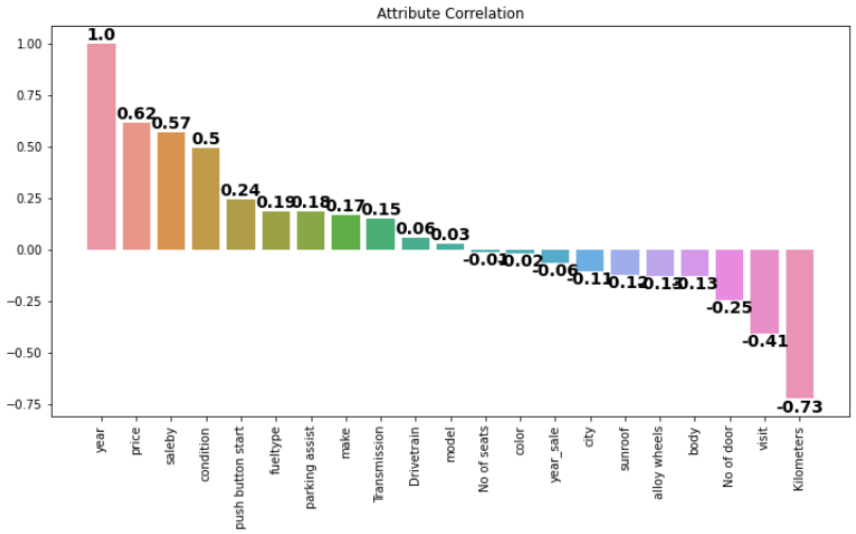
Se analizó el índice de correlación de los atributos dependientes contra el precio logrando el siguiente resultado:



Identificando como columnas o atributos dependientes:
x: city, saleby, make, model , condition, body, fueltype, color, year_sale, No of door , visit, year, Kilometers, Transmission, push button start, parking assist, sunroof, alloy wheels.

Modelo de Clasificación:
Se determinó que para el modelo de clasificación se predecirá el año del modelo del auto (atributo target **y = year**).

Se analizó el índice de correlación de los atributos dependientes contra el año logrando el siguiente resultado:



Identificando como columnas o atributos dependientes:
x: price, Kilometers, condition, saleby.

La división entre train set y test set se efectuó de 80% y 20% respectivamente

c) Ejecución y validación de modelos

c.1) Modelos de regresión

Para los modelos de regresión se usaron 5 algoritmos diferentes: Regresión lineal, regresión polinomial , Random Forest, Tree Regression y Gradient Boosting Regressor

En general, para aplicar cualquiera de estos modelos, es necesario definir nuestra columna objetivo y los atributos dependientes:

```
[ ]  # Se asigna el target "price"
     y = np.array(df['price'])

[ ]  # Se asignan las variables dependientes
     x = np.array(df[['city','saleby','make','model','condition','body','fueltype','color','year_sale','No of door','visit','year','Kilometers','Transmission','push button start','parking assist','sunroof','alloy wheels'] ])
```

De esta forma dispondremos de una columna Y y un conjunto de atributos X. Luego es necesario separar el dataset en un conjunto de entrenamiento y otro de prueba, para todos los casos se tomó un 80% del set para entrenar y 20% para probar.

```
# Separacion grupos de datos, se instancia el regresor lineal, se entrena y se predice
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
```

Cabe destacar que, en la aplicación de diferentes modelos, usualmente lo que más varía es el objeto a instanciar (del módulo de la librería que se encargará del entrenamiento) y los parámetros. En la figura de arriba se aprecia la aplicación de la regresión lineal.
Luego se ejecuta el bloque de comandos el cual entrenará nuestro modelo de aprendizaje. Posterior a ello, se disponen de herramientas para comprobar la eficiencia y precisión de nuestro modelo, la principal es el score de precisión:

```
print('Precisión Regresión Lineal:')
print(regressor.score(X_train, y_train))
```

```
Precisión Regresión Lineal:
0.4796795499579203
```

En este caso la precisión de nuestro modelo es de 47% lo cual no es un valor positivo, ya que se espera que la precisión sea un valor mayor a 90% - 95%.
De esta manera se disponen de 5 métricas las cuales también evaluarán el desempeño del modelo:

```
#METRICAS ADICIONALES PARA LA REGRESION LINEAL
print('1. Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('2. Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('3. R2 Score:', (metrics.r2_score(y_test,y_pred)))
print('4. Max Error:', ( max_error(y_test, y_pred)))
print('5. Explained variance score: ',(explained_variance_score(y_test,y_pred)))
```

```
1. Mean Absolute Error: 12178.26359822009
2. Mean Squared Error: 293426106.5227198
3. R2 Score: 0.471854976446184
4. Max Error: 76337.02050374704
5. Explained variance score:  0.4718614088946058
```

Los algoritmos con mejor desempeño para el modelo de regresión resultaron ser:

Random Forest, con una precisión del **95,49%**

```
MAE- Dropeo:
Precisión del modelo
0.95498045463774
METRICAS PARA RANDOM FOREST
1. Mean Absolute Error: 7227.403286576015
2. Mean Squared Error: 137501842.51047745
3. R2 Score: 0.7518054759059772
4. Max Error: 81285.3
5. Explained variance score:  0.7519085745136833
7227.403286576015
```

Decision Tree Regression, con una precisión del **99,79%**

```
Precisión del modelo:
0.9979008374067228

 METRICAS ADICIONALES PARA TREE REGRESSION
 1. Mean Absolute Error: 8937.182046858012
 2. Mean Squared Error: 224112148.822005
 3. R2 Score: 0.5947099539979483
 4. Max Error: 99488.0
 5. Explained variance score:  0.5947565969301234
```

Este último modelo podría estar sobre-ajustado puesto que se verificó que para "algunos" valores nuevos se predicen valores que podrían no estar de acuerdo a la realidad, por lo que una alternativa muy confiable es el **Random forest** pese a tener un **95%** de precisión en comparación al **99%** del **Decision Tree**

Para el resto y en general todos los modelos analizados e implementados , el detalle completo y extenso tanto del código como de los resultados se encuentran en el archivo de Colab.

c.2) Modelos de clasificación

Para los modelos de clasificación se usaron 5 algoritmos diferentes: Regresión logística, SVM, K-Neighbors, Decision Tree y Random Forest
Los pasos son similares, solo que nuestro target cambia y también los atributos:

```
# Se asigna el target y los atributos dependientes

y = np.array(df['year'])
x = np.array(df[['price','Kilometers','condition','saleby'] ])

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

##Regresión logística
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
Y_pred = logreg.predict(X_test)
print('Precisión Regresión Logística:')
print(logreg.score(X_train, y_train))
```

En este caso se muestra el uso de la regresión logística. Para su evaluación se realiza el mismo cálculo de precisión y métricas.

```
Precisión Regresión Logística:
0.3307716273756361
```

```
#METRICAS ADICIONALES PARA REGRESION LOGISTICA
print('METRICAS PARA REGRESION LOGISTICA')
print('1. Mean Absolute Error:', (metrics.mean_absolute_error(y_test,Y_pred)))
print('2. Mean Squared Error:', metrics.mean_squared_error(y_test,Y_pred))
print('3. R2 Score:', (metrics.r2_score(y_test,Y_pred)))
print('4. Max Error:', ( max_error(y_test,Y_pred)))
print('5. Explained variance score: ',(explained_variance_score(y_test,Y_pred)))
```

```
METRICAS PARA REGRESION LOGISTICA
1. Mean Absolute Error: 2.354346245716066
2. Mean Squared Error: 18.341468480631427
3. R2 Score: 0.4533059250088529
4. Max Error: 31
5. Explained variance score:  0.49826564928429995
```

El algoritmos con mejor desempeño para el modelo de clasificación resultó ser:

Decision Tree Classifier, con una precisión del **99,59%**

```
Precisión Árboles de Decisión Clasificación: 0.9959497351749922


1. Mean Absolute Error: 2.3263059507737043
2. Mean Squared Error: 16.21102918267733
3. R2 Score: 0.510352409221718
4. Max Error: 28
5. Explained variance score:  0.5103829129914559
```

Para el resto y en general todos los modelos analizados e implementados , el detalle completo y extenso tanto del código como de los resultados se encuentran en el archivo de Colab.

d) Predicciones

Para las predicciones, ya se tienen entrenados los modelos y ya solo se necesita la interacción del usuario para predecir nuevos datos introducidos por el mismo:

```
print('Ingresar datos para predecir precio:')

city = 13
while city > 12 or city <0:
  entrada = int(input("Introduzca un dato de ciudad entre 0 y 12: "))
  if entrada < 13 and entrada > -1:
    city = entrada
    break
```

Las entradas se pueden ingresar mediante línea de comandos y están preestablecidas en un rango claramente definido. Posterior a ello los resultados se cargan a una lista que luego serán el nuevo conjunto de pruebas para las predicciones.

```
testing = [city,saleby,make,model,condition,body,fueltype,color,year_sale,door,visit,year,kilometers,transmission,pushbutton,parking,sun,alloy ]
```

```
# Librerías necesarias para el algortimos Random forest Regressor
from sklearn.ensemble import RandomForestRegressor

y = np.array(df['price'])
x = np.array(df[['city','saleby','make','model','condition','body','fueltype','color','year_sale','No of door','visit','year','Kilometers','Transmission','push button start','parking assist','sunroof','alloy wheels'] ])

X_train_p, X_test_p, y_train_p, y_test_p = train_test_split(x, y, test_size=0.2)

model = RandomForestRegressor(n_estimators=10)
model.fit(X_train_p,y_train_p)
preds_val = model.predict(np.array(testing).reshape(1,-1))
print('Valor predecido con Randon Forest Regressor:')
print(preds_val)
```

```
Valor predecido con Randon Forest Regressor:
[45656.2]
```

Esta interacción se puede realizar tanto con los modelos de regresión como con los de clasificación, dependiendo del grado de éxito que sea obtenido.

5. Conclusiones

Como resultado del proyecto se obtuvieron las siguientes conclusiones:

Para el modelo de regresión, se encontró que el algoritmo de Árbol de decisión para regresión es el que presenta mayor precisión para el tipo de problema de predicción del precio de venta de un vehículo, tomando en consideración los atributos dependientes y relevantes del dataset provisto.

Para el modelo de regresión, otros algoritmos como el de regresión lineal y regresión polinomial, no presentaron una buena precisión debido a la no existencia de un índice de correlación elevado entre los diferentes atributos, ningún caso presentaba una tendencia lineal y si bien en alguna correlación se veía una tendencia polinomial, la misma no se podía ajustar.

Para el modelo de regresión, el algoritmo de Random Forest para regresión presenta también una buena precisión, para el tipo de problema de predicción del precio de venta de un vehículo, tomando en consideración los atributos dependientes y relevantes del dataset provisto, por lo que se considera con una segunda opción de solución.

Para el modelo de clasificación, se encontró que el algoritmo de Decision Tree es el que presenta mayor precisión para el tipo de problema de clasificación del año del modelo de un vehículo, tomando en consideración los atributos dependientes y relevantes del dataset provisto.

Para el modelo de clasificación, otros algoritmos como el de regresión logística, k-neighbors, SVM o Random Forest no presentaron una buena precisión debido a la no existencia de un índice de correlación elevado entre los diferentes atributos.

Durante la elaboración del proyecto, pudimos constatar que la fase más compleja fue la de pre-procesamiento de la información, puesto que algún caso existieron muchos datos perdidos los cuales no necesariamente deberían ser eliminados, más por el contrario se trata de imputar los mismos con los valores de la media, la moda o en algún caso incluso no se llegó a utilizar un valor estadístico, sino que se utilizó un valor subjetivo aplicado al contexto del problema de venta de autos de acuerdo a su mayor probabilidad, por ejemplo el ítem "parqueo asistido" no es valor común, por lo que los nulos se rellenaron con el valor contrario, es decir que no dispone de esa característica.

Para el cambio de valores categóricos a valores numéricos por lo general se utilizó una regla de secuencia de asignación para los valores numéricos, puesto que no se identificó que corresponda un peso ordinal a dichos valores.

Se encontraron varias columnas completamente irrelevantes para el análisis del problema.

No se encontraron índices de correlación bastante fuertes, los mejores índices solo bordeaban el 0.5.
Se encontraron muchos valores en distintas columnas que eran outliers (valores atípicos o extremos) correspondientes a errores en la información, como también valores extremos reales correspondientes a excepciones, las cuales tuvieron que de igual forma ser eliminadas puesto que afectan a la distribución de los datos.
No se encontraron índices de correlación bastante fuertes, los mejores índices solo bordeaban el 0.5.
Para la división de los datos en entrenamiento y pruebas se utilizó una relación 80% / 20% esto de acuerdo a las prácticas revisadas en ejemplos de estudio.

6. Bibliografía

-Aurelian Geron, Hands-on Machine Learning.
-Tom Mitchell, Machine Learning.
-Documentación de ayuda libreria "Sklearn"
-Documentación de ayuda libreria "Pandas"
-Documentación de ayuda librería "Matlib"
-https://www.youtube.com/c/AprendeIAconLigdiGonzalez
-https://www.youtube.com/c/codificandobits