Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero Pérez	23/01/2023
	Nombre: Juan carlos	

Actividad: Árboles y *random forest* para regresión y clasificación

Introducción

A lo largo del nuestro día se toman variadas decisiones, si vamos al trabajo en camión o en autobús, si desayunamos o no desayunamos, son interminables estas elecciones para llegar a algún resultado en concreto.

En la inteligencia artificial se usan arboles de decisiones los cuales replican al ser humano al momento de la toma de decisiones y mediante estos arboles se pueden realizar predicciones eficientes con los datos proporcionados por el usuario, estos arboles se dividen en 2, regresión y clasificación. Además, se cuenta con los bosques aleatorios, que basan su funcionamiento en varios árboles de decisión para mejorar los resultados, es decir si hay 100 arboles en base a los resultados que arroje cada árbol es que se tomara la decisión al momento de arrojar alguna predicción.

El objetivo de esta práctica es conocer mejor como funcionan estos algoritmos para realizar predicciones en el precio de viviendas, utilizando un set de datos tomado de la página de kaggle con múltiples datos recolectados.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero Pérez	23/01/2023
	Nombre: Juan carlos	

Desarrollo

Árbol de Regresión

La primera parte de la practica se centra en los árboles de decisión, los cuales son una técnica de aprendizaje supervisado, es decir en este caso el árbol utiliza técnicas de regresión para tomar las decisiones pertinentes y realizar la predicción adecuada. Con la regresión se busca predecir el valor de una variable continua.

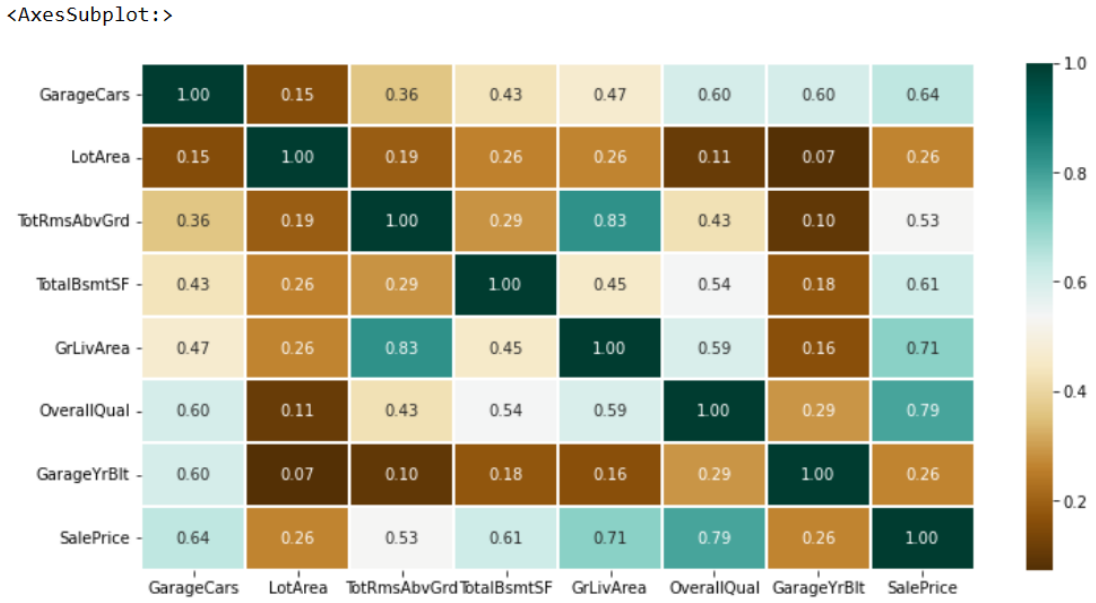
Para la primera parte la práctica, se seleccionaron las siguientes variables, basándose mayormente en investigación y posibles conjeturas de las principales variables que podrían influir dentro del precio a futuro de una casa.

- GarageCars . – Se tomo esta variable debido a que se da por hecho que una casa la cual cuenta con garage para carros esta ligada con el precio de esta.
- LotArea . – Se tomo esta variable debido a que se considera que el tamaño de la casa es algo importante.
- TotRmsAbvGrd. – Se asigno esta variable al ejercicio debido a la importancia de las habitaciones en la casa.
- TotalBsmtSF. – Se selecciono debido a que el tamaño del sótano y si cuenta con alguno, influye en el precio.
- GrLivArea.- Esta variable se refiere a los pies cuadrados sobre la superficie, por lo cual se eligió.
- OverallQual. – Se refiere al material general y acabado de la vivienda, por lo cual se intuye influye en el precio.
- GarageYrBlt. – Se refiere al año de construcción del garage y año de construcción de la vivienda, además de traer datos nulos, los cuales nos servirán para el tratamiento del problema.
- SalePrice. – Se refiere a la variable objetivo (Precio de las viviendas).

Se identifican las variables correlacionadas con la variable de interés, mediante una matriz de correlación.

Figura 1.-

Matriz de correlación para árbol de regresión



En la matriz de correlación anterior, se puede observar que una de las variables con alta dependencia a la variable objetivo es overralQual, la cual tiene un 0.79 de relación con la variable SalePrice.

Para el tratamiento de los valores nulos en este caso, la única variable que presenta valores nulos es el garage, por lo que se entiende que estos representan las viviendas que no cuentan con un garage alguno y se decidió aplicar un rellenado de celdas a todos esos valores con una cifra de 0.

Después de realizar un tratamiento a los valores de missing, se divide el set de datos en 2, una variable “X”, donde estarán todas las variables menos la variable objetivo y una variable “y”, la cual contendrá únicamente la variable objetivo.

```
X = dfFilter1.iloc[:, :-1]
y = dfFilter1.iloc[:, -1]
```

Una vez realizado esto, con la función train_test_split dividimos nuestros sets de datos en pequeñas fracciones, una para que nuestro modelo realice entrenamiento y otra como test.

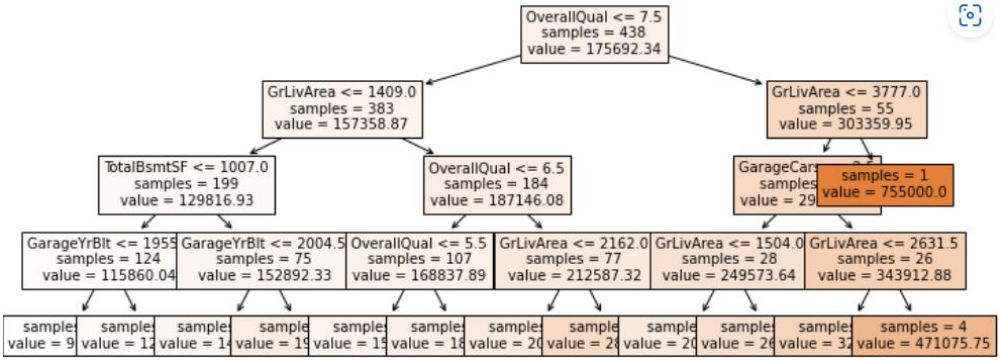
Realizamos nuestro árbol con la función DecisionTreeRegressor y para entrenar nuestro modelo utilizamos la función fit y le pasamos como parámetros los sets de entrenamiento. Para finalmente visualizar el árbol se utiliza la siguiente función:

Figura 2.-

Visualización del árbol de regresión

```
from sklearn.tree import plot_tree
fig, ax = plt.subplots(figsize=(12, 5))
plot = plot_tree(
            decision_tree = modelo,
            feature_names = dfFilter1.drop(columns = "SalePrice").columns,
            class_names   = 'SalePrice',
            filled        = True,
            impurity      = False,
            fontsize      = 10,
            precision     = 2,
            ax            = ax
       );

plot
```



Este es un árbol de decisión y para interpretarlo se empieza con la primer variable, por ejemplo de la primera variable que es “OverallQual” elige el camino, si es menor o igual a 7.5, se elegirá el camino de la izquierda y así seguirá todo el proceso hasta terminar dando con la predicción del precio final.

Para evaluar el desempeño de nuestro modelo se pueden utilizar el Error Cuadrático Medio, el cual nos sirve para ver el error que presentan los datos, es decir entre mas cercano al 0 mejor será nuestro modelo y el coeficiente de determinación-R2, el cual nos sirve para evaluar el rendimiento de nuestro modelo.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero Pérez	23/01/2023
	Nombre: Juan carlos	

Para el primer ejercicio se dieron los siguientes resultados:

```
y_pred = tree.predict(X)
sqrt(mean_squared_error(y,y_pred))
```

```
43173.85131503983
```

```
r2_score(y, y_pred)
```

```
0.7044484031535154
```

Como se puede observar, los resultados no fueron buenos, asi que se decidio realizar un análisis para ver cuales son las variables que mas tomaba en cuenta el modelo.

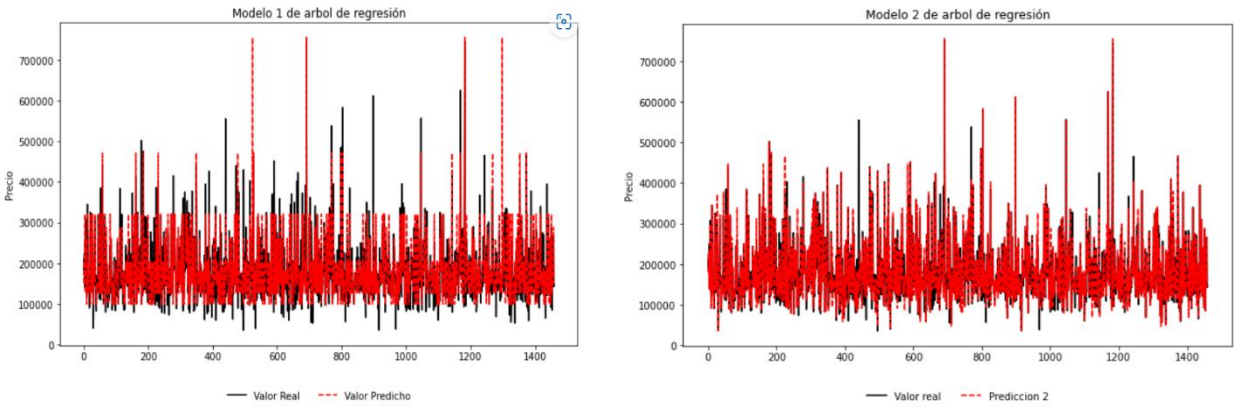
Figura 3.-

Importancia de los predictores en el modelo

	predictor	importancia
5	OverallQual	0.559496
4	GrLivArea	0.331953
0	GarageCars	0.059503
3	TotalBsmtSF	0.031784
6	GarageYrBlt	0.017264
1	LotArea	0.000000
2	TotRmsAbvGrd	0.000000

Analizando la tabla, los 3 valores principales que más toma en cuenta nuestro modelo son "TotalBsmtSF","GrLivArea" y "OverallQual". Por lo cual se planteo la siguiente pregunta ¿Que pasa si reducimos nuestras variables a las 3 principales predictores?. Una vez planteada la pregunta, se creo un nuevo dataFrame con las 3 principales variables y se realizo todo el proceso para crear un nuevo model, los resultados fueron los siguientes:

Figura 4.- *Graficas de predicción modelo 1 vs modelo 2*



Observando las graficas se llega a la conclusión de que el modelo 2 presenta mejores rendimientos en cuanto a predicciones y esto se verifica con los datos obtenidos para el Error Cuadrático Medio y el coeficiente de determinación-R2.

```
print("El error cuadrado medio para el modelo 2 es: ",sqrt(mean_squared_error(y2, y_pred2)))
print("El coeficiente de Determinación para el modelo 2 es: ",r2_score(y2, y_pred2))
```

```
El error cuadrado medio para el modelo 2 es:  24403.601851766245
El coeficiente de Determinación para el modelo 2 es:  0.9055722614956586
```

Con esto obtenemos que nuestro modelo alcance un menor error y un mejor coeficiente de determinación.

Random Forest de Regresión

Para abordar este primer modelo de random forest, se utilizó el mismo set de datos que el del primer modelo del árbol de regresión y se realiza un proceso de entrenamiento y división de datos similar al del árbol de regresión, uno de los cambios más importantes es que con la función RandomForestRegressor crearemos nuestro bosque aleatorio y mediante el parámetro n_estimators le indicaremos el número de árboles que tendrá nuestro bosque, para este caso se realizó con 100 y se obtuvieron los siguientes resultados:

```
print("El error cuadrado medio para el modelo 1 de RF es: ",sqrt(mean_squared_error(y, y_pred3)))
```

```
El error cuadrado medio para el modelo 1 de RF es:  29713.091582697285
```

```
print("El error coeficiente de determinación para el modelo 1 de RF es: ",r2_score(y, y_pred3))
```

```
El error coeficiente de determinación para el modelo 1 de RF es:  0.8600130954995602
```

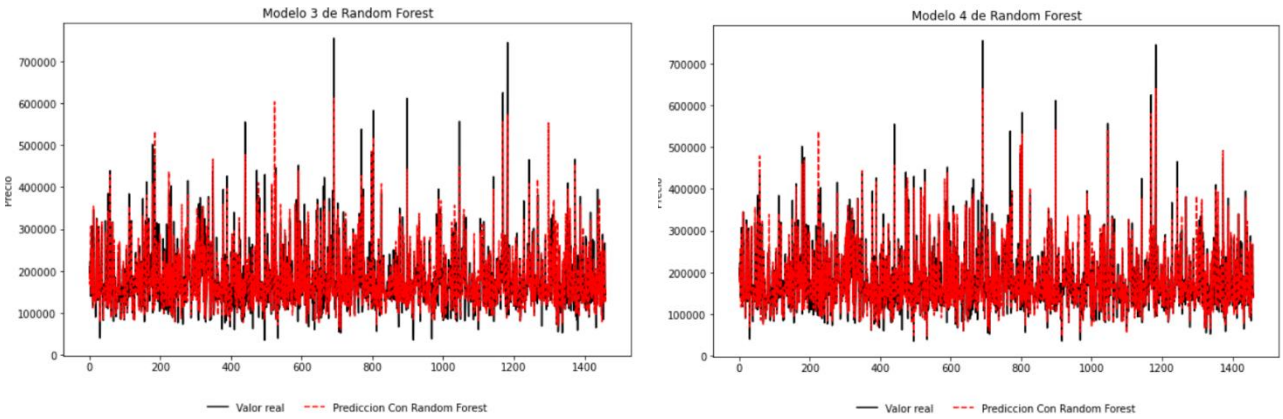
Comparado con nuestro primer modelo, mejoran notablemente los resultados para el set de datos extenso, se realizó el mismo proceso, pero con el set de datos utilizado para el modelo 2 del árbol de regresión y con una cantidad de 1000 árboles para nuestro bosque y se obtuvieron los siguientes resultados:

```
print("El error cuadrado medio para el modelo 2 de RF es:",sqrt(mean_squared_error(y2, y_pred4)))
print("El coeficiente de Determinación para el modelo 2 de RF es:",r2_score(y2, y_pred4))
```

```
El error cuadrado medio para el modelo 2 de RF es: 21568.843290147262
El coeficiente de Determinación para el modelo 2 de RF es: 0.9262358338829325
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero Pérez	23/01/2023
	Nombre: Juan carlos	

Figura 5.- *Graficas de predicción modelo 3 vs modelo 4*



Observando los datos y las gráficas se puede determinar que con un set de datos reducido y con random forest se puede llegar a crear un modelo adecuado para realizar predicciones.

Árbol de Clasificación

Para la práctica de clasificación se seleccionaron variables categóricas, las cuales son las siguientes:

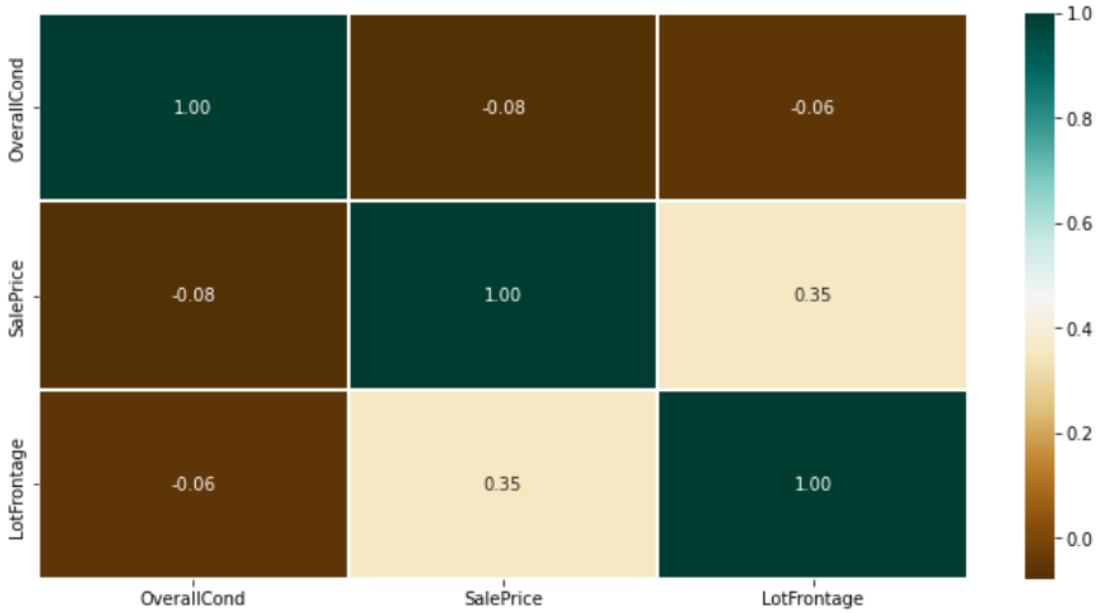
- LotConfig. – Se refiere a configuración de la casa.
- BldgType. – Con esta variable se tomó en cuenta el tipo de vivienda, si está hecha para una solo familia, entre otras categorías.
- OverallCond.- Se refiere al estado en el que se encuentra la vivienda.
- Exterior1st. – Esta variable proporciona el material de revestimiento exterior de la vivienda.
- LotFrontage.- Hace referencia a los pies lineales de calle conectados a la propiedad.
- SalePrice. - Se refiere a la variable objetivo (Precio de las viviendas).

Una vez definidas las variables, se realizó un conteo para identificar con que tipos de datos se estará trabajando.

```
Variables Categoricas: 3
variables Enteras: 2
Variables Flotantes: 1
```

Una vez identificado las variables y su tipo, se realizó una matriz de correlación para observar las variables más dependientes del objetivo.

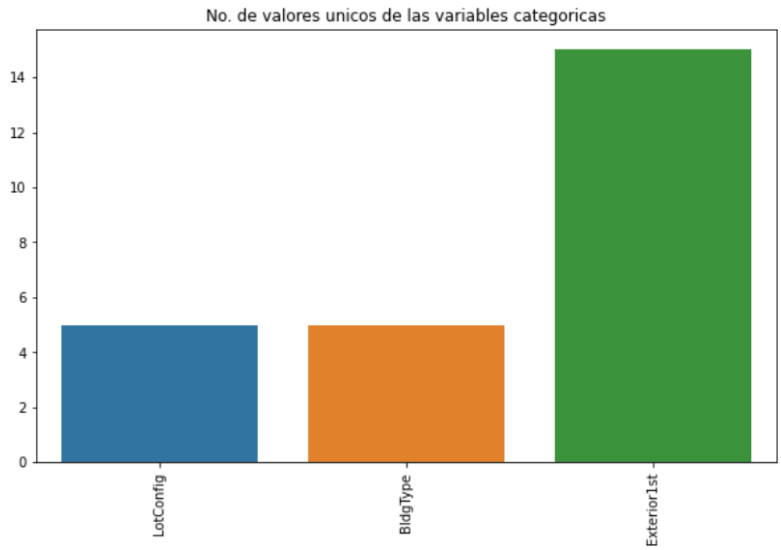
Figura 6.- *Matriz de correlación para árbol de clasificación*



Con esta matriz se puede determinar que una de las variables más relacionadas con el precio es LotFrontage.

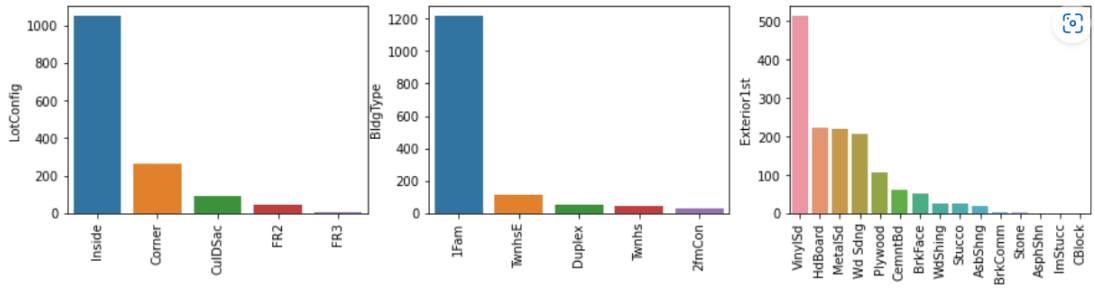
Una vez obtenido esto, se analizan las variables categóricas, haciendo gráficos podemos observar la cantidad de categorías de cada variable.

Figura 7.- *Grafico de cantidad de categorías por variable*



Con el grafico se puede determinar que la variable con más categorías es "Exterior1st", la cual cuenta con aproximadamente 15 datos categóricos, si se quiere ejemplificar más a fonde, se realiza un gráfico para cada variable y así observar el nombre de cada una de ellas.

Figura 8.- *Grafico desglosado de cantidad de categorías por variable*



Con estos gráficos obtenemos la información de cada una de las variables categóricas, se procede a analizar los datos faltantes o nulos:

```
dfFilter1.isnull().sum()
```

```
LotConfig        0
BldgType         0
OverallCond      0
Exterior1st      0
SalePrice        0
LotFrontage    259
dtype: int64
```

Como se observa, solo son 259 datos nulos para nuestro set de datos, por lo que se decidió utilizar la media para rellenar estos espacios vacíos.

Una vez decidido que se haría con los valores nulos, se aplicó una función llamada dummies para realizar una conversión de los datos categóricos a binarios, es decir se dividirá cada categoría y se asignara 1 o 0 en caso de ser verdadero o falso, esto se realiza para que Python pueda interpretar estos datos.

```
df = pd.get_dummies(data = dfFilter1,drop_first = True)
```

El proceso para realizar el modelo es muy similar con el de regresión, se crea una variable "X", donde estarán todas las variables menos la objetivo y una "y", donde estará únicamente la variable objetivo.

Posteriormente con la siguiente librería y función, se crea el modelo y se entrena con los datos.

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=2,criterion='gini')
modelo1 = tree.fit(X_train,y_train)
```

Para medir que tan bueno es nuestro modelo se utiliza la función accuracy, la cual mide el porcentaje de aciertos que ha tenido nuestro modelo y también se mide la puntuación de los 2 sets de datos, se obtuvieron los siguientes resultados con este modelo.

```
Accuracy: 0.2237442922374429
puntuación del modelo en los datos de entrenamiento: 0.2237442922374429
puntuación del modelo en los datos de testing: 0.00821917808219178
```

Los resultados obtenidos fueron malos, como se observa, solo hay un 22% de probabilidad de que acierte en los resultados, por lo que se decidió reducir el set de datos y sacar a la variable categórica que mas categorías tiene, la cual es “Exterior1st”.

Nuevamente se crea el modelo, con una profundidad de 20 y un criterio de entropy, se calcula el accuracy, obteniendo los siguientes resultados:

```
y_trainOut2=modelo2.predict(X_train2)
print("Accuracy:",metrics.accuracy_score(y_train2, y_trainOut2))
print(f'puntuación del modelo en los datos de entrenamiento: {modelo2.score(X_train2, y_train2)}')
print(f'puntuación del modelo en los datos de testing: {modelo2.score(X_test2, y_test2)}')
```

```
Accuracy: 0.4041095890410959
puntuación del modelo en los datos de entrenamiento: 0.4041095890410959
puntuación del modelo en los datos de testing: 0.003424657534246575
```

Como se observa, los datos mejoran, sin embargo, nuestro modelo sigue siendo deficiente a la hora de predecir.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero Pérez	23/01/2023
	Nombre: Juan carlos	

Random Forest de Clasificación

Para realizar este modelo, se decidió reducir el set de datos, dejando las 3 principales variables y reduciendo el tamaño del set.

```
dfFilter3 = dfFilter2[["BldgType","SalePrice","LotFrontage"]]
df3 = pd.get_dummies(data = dfFilter3,drop_first = True)
df_final = df3.sample(frac=1/10,replace=True)
df_final
```

El proceso de creación del modelo es similar al de regresión, se dividen las variables en 2 y se utiliza test_split para dividir nuestros sets en entrenamiento y testeo. Para este caso en particular se usaron 100 árboles de clasificación y se utilizó el criterio Gini.

```
from sklearn.ensemble import RandomForestClassifier

X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3,test_size=0.25, random_state=0)
forest=RandomForestClassifier(n_estimators = 100,
                              criterion = 'gini',
                              max_features = True,
                              max_samples = 1/10,
                              oob_score=True)

modelo3 = forest.fit(X_train3,y_train3)
```

Los resultados obtenidos con este modelo fueron los siguientes:

```
y_trainOut3 = modelo3.predict(X_train3)
print("Accuracy:",metrics.accuracy_score(y_train3, y_trainOut3))
print(f'puntuación del modelo en los datos de entrenamiento: {forest.score(X_train3, y_train3)}')
print(f'puntuación del modelo en los datos de testing: {forest.score(X_test3, y_test3)}')
```

```
Accuracy: 0.24770642201834864
puntuación del modelo en los datos de entrenamiento: 0.24770642201834864
puntuación del modelo en los datos de testing: 0.02702702702702703
```

Como se observa los resultados siguen siendo malos para este problema de predicción.

Para evaluar el desempeño también se creó una matriz de confusión, sin embargo, debido a la cantidad de datos no es posible visualizar algún dato bien.

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero Pérez	23/01/2023
	Nombre: Juan carlos	

Árbol de Clasificación parte 2

Se realizo un ajuste de modelo de ultima hora, al cual se le agrego una condición para clasificar mejor, se agrupo por precio bajo menor o igual a 100 000, medio mayor a 100 001 y menor a 500 000 o alto, mayor a 500 000, con esta condición, lo agregamos al dataFrame y se le asigno la etiqueta “catPrecio”.

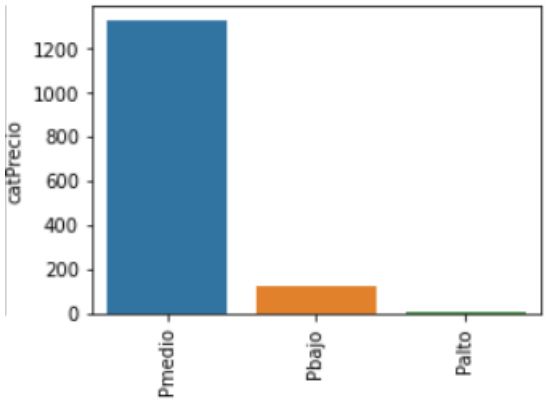
```
condicion = [
    (dfTrain['SalePrice']<= 100000),
    (dfTrain['SalePrice']> 100000) & (dfTrain['SalePrice']<= 500000),
    (dfTrain['SalePrice']> 500000)
]
valor = ['Pbajo','Pmedio','Palto']
dfTrain['catPrecio'] = np.select(condicion,valor)
```

Se realizo todo el proceso de análisis de variables utilizados con el algoritmo de clasificación anterior.

Para asignarle una etiqueta a las categorías de clasificación, se utilizó la librería LabelEncoder, y se asignaron estos valores en la variable “catPrecioLabel”.

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
dfFilter1['catPrecioLabel'] = le.fit_transform(dfFilter1['catPrecio'])
```

Figura 9.- *Visualización de las distintas clasificaciones para la variable catprecio*




```
dfFilter1['catPrecioLabel'].value_counts()
```

```
2    1328
1     123
0       9
Name: catPrecioLabel, dtype: int64
```

Con la función value_counts y la gráfica donde se muestran las categorías para cada variable, se puede identificar qué precios bajos se asignó el número 1 y solo hay 123, medios se asignó el número 2 y cuenta con 1328, mientras que los precios altos tienen el numero 0 y en total se cuenta con 9.

Se realizo el mismo proceso de separar los datos en 2, esta vez dejando en la variable “y” la etiqueta “catPrecioLabel”, la cual será la variable objetivo para determinar si el precio de una casa es bajo, medio o alto y se separó con la función test_split, además de crear un árbol con una profundidad de 10 y utilizando el criterio de Gini.

Una vez realizado esto, se entrena el modelo y se saca el accuracy y las métricas para ver el desempeño del modelo.

```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_train, y_trainOut))
print(f'puntuación del modelo en los datos de entrenamiento: {modelo1.score(X_train, y_train)}')
print(f'puntuación del modelo en los datos de testing: {modelo1.score(X_test, y_test)}')
```

```
Accuracy: 1.0
puntuación del modelo en los datos de entrenamiento: 1.0
puntuación del modelo en los datos de testing: 1.0
```

Viendo los resultados se observa un modelo que predice perfectamente el precio de una casa. Se puede visualizar mejor con la siguiente tabla.

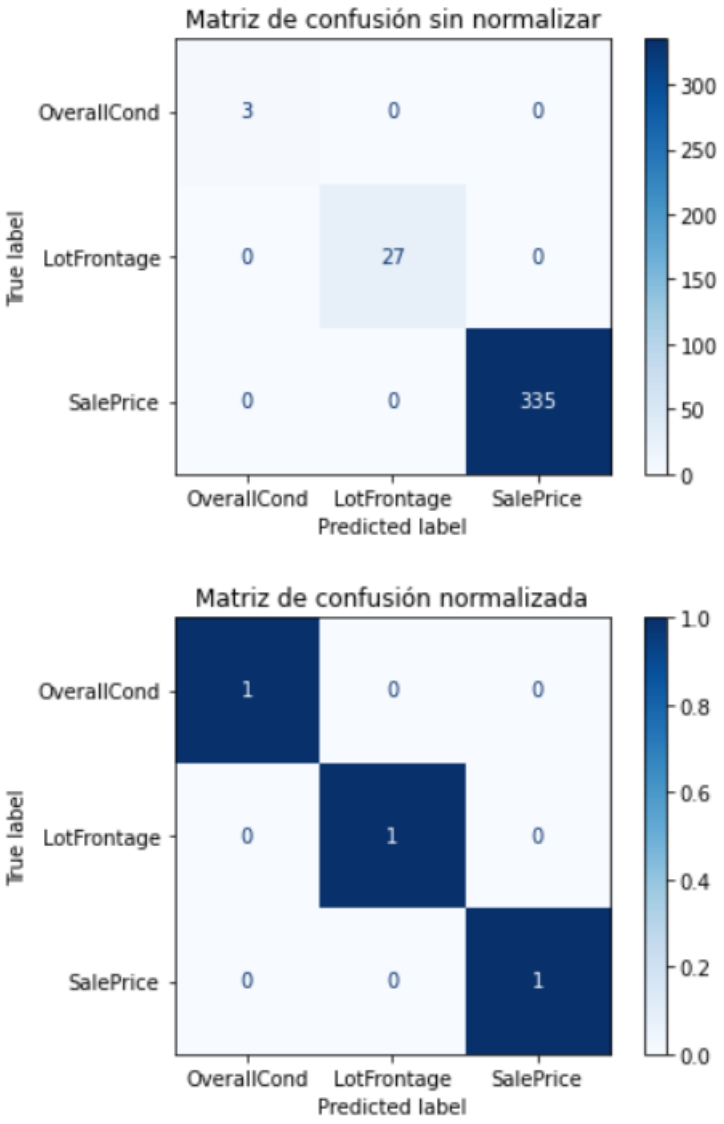
Figura 10.- *Tabla de predicción del precio mediante clasificación*

	SalePrice	Prediccion del precio
768	216837	2
481	374000	2
1035	84000	1
1437	394617	2
45	319900	2
1234	130000	2
458	161000	2
400	245500	2
684	221000	2
1206	107000	2

Interpretando la tabla se observa que predijo correctamente el precio de todos, por ejemplo, del valor 216 837 lo predijo en un dos lo cual significa que se encuentra con un precio medio, debido a que esta en un rango mayor a 100,000 y menor a 500,00. Mientras que para el precio de 84000 lo predijo con un valor de 1, lo cual significa que es un precio bajo, esto debido a que se encuentra debajo de los 100,000.

Por último, para comprobar que nuestro modelo si presenta un 100% de acertamiento se realizo una matriz de confusión.

Figura 11.- *Matriz de confusión*



Como se muestra en la matriz, todos se encuentran en diagonal por lo cual no presenta ningún falso negativo o falso positivo, por lo que se puede decir que este modelo cumple con su funcionamiento de predicción del precio.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero Pérez	23/01/2023
	Nombre: Juan carlos	

Conclusiones

Una de los obstáculos más grande que presente al momento de realizar esta práctica, es la interpretación de los datos y se puede observar muy bien con la primera parte de clasificación, la cual estaba intentando entrenar al modelo sin realizar agrupaciones y fue hasta el último momento en el cual me percate de esto y realice otro archivo para no afectar el ya hecho, el cual quise dejar en la práctica para dejar claro que se puede aprender de los errores, obteniendo mejores resultados de lo esperado, cambiando el punto de vista y análisis.

Analizando todos los modelos realizados el mejor modelo a la hora de predecir el precio con valores numéricos, fue el modelo 2 de Random forest de regresión, el cual tuvo una efectividad del 92% a la hora de predecir resultados. Sin embargo, el último modelo de árbol de clasificación realizado, presento un 100% por lo cual se podría decir que es perfecto para su cometido, no obstante, este tipo de modelos no predice el precio exacto de la vivienda, sino que predice un rango en el cual se encuentra dicho precio.

En mi opinión, ambos modelos son buenos y vale la pena profundizar mas en el tema, utilizar algunas técnicas de podado o técnicas de ajuste a nuestro modelo para así tener mejores resultados.