

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

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

### Librerías a utilizar

```
In [ ]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, export_graphviz
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor
from sklearn.tree import plot_tree
#Visualizacion
import matplotlib.pyplot as plt
import seaborn as sns
import graphviz
%matplotlib inline
#metricas
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn import svm
from sklearn import metrics
from funpymodeling.exploratory import freq_tbl, profiling_num
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
```

### Uso de las librerías

Sklearn fue la opción que se decidió usar, ya que en las practicas que se realizaron para cada tema de la materia, se uso la librería.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

## Lectura y primer tratamiento de datos

```
In [ ]: #Leer los datos de los archivos train y test
data_train = pd.read_csv("housing_train.csv")
data_test = pd.read_csv("housing_test.csv")
#se concatenan los 2 archivos para su mejor análisis.
frames = [data_train,data_test]
data = pd.concat(frames)
#sacamos el promedio de los precios de venta para llenar los Null que existan.
prom = data["SalePrice"].mean()
data["SalePrice"].fillna(prom , inplace = True)
data_c = data
```

Se decidió que los valores en blanco de la columna SalePrice debían rellenarse con el promedio de los demás valores, ya que esa variable es indispensable al momento de hacer nuestro modelo de regresión y el promedio se acerca un poco más a la vida real de los precios valorados de inmuebles.

## Primeros Análisis de datos

```
In [4]: #analisis estadísticos de variables numéricas
data_numericos = profiling_num(data)
data_numericos
```

Con la librería **funpymodeling**, se pueden obtener fácilmente un análisis de los datos numéricos y categóricos (el detalle completo del análisis se encuentra en el código fuente anexo a este trabajo)

## Datos numéricos

	variable	mean	std_dev	variation_coef	p_0.01	p_0.05	p_0.25	p_0.5	p_0.75	p_0.95	p_0.99
0	Id	1460.000000	842.787043	0.577251	30.18	146.90	730.5	1460.000000	2189.500000	2773.10	2889.82
1	MSSubClass	57.137718	42.517628	0.744125	20.00	20.00	20.0	50.000000	70.000000	160.00	190.00
2	LotFrontage	69.305795	23.344905	0.336839	21.00	32.00	59.0	68.000000	80.000000	107.00	135.68
3	LotArea	10168.114080	7886.996359	0.775660	1680.00	3182.00	7478.0	9453.000000	11570.000000	17142.90	33038.64
4	OverallQual	6.089072	1.409947	0.231554	3.00	4.00	5.0	6.000000	7.000000	8.00	10.00
5	OverallCond	5.564577	1.113131	0.200039	3.00	4.00	5.0	5.000000	6.000000	8.00	9.00
6	YearBuilt	1971.312778	30.291442	0.015366	1900.00	1915.00	1953.5	1973.000000	2001.000000	2007.00	2008.00
7	YearRemodAdd	1984.264474	20.894344	0.010530	1950.00	1950.00	1965.0	1993.000000	2004.000000	2007.00	2009.00
8	MasVnrArea	102.201312	179.334253	1.754716	0.00	0.00	0.0	0.000000	164.000000	466.50	771.05
9	BsmtFinSF1	441.423235	455.610826	1.032141	0.00	0.00	0.0	368.500000	733.000000	1274.00	1635.32
10	BsmtFinSF2	49.582248	169.205611	3.412625	0.00	0.00	0.0	0.000000	0.000000	435.00	874.66

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

## Datos Categóricos

```
In [5]: #Análisis estadísticos de datos categoricos
```

```
data_categoricos= freq_tbl(data)
```

	MSZoning	frequency	percentage	cumulative_perc
0	RL	2265	0.775951	0.777015
1	RM	460	0.157588	0.934820
2	FV	139	0.047619	0.982504
3	RH	26	0.008907	0.991424
4	C (all)	25	0.008565	1.000000

	Street	frequency	percentage	cumulative_perc
0	Pave	2907	0.995889	0.995889
1	Grvl	12	0.004111	1.000000

	Alley	frequency	percentage	cumulative_perc
0	Grvl	120	0.041110	0.606061
1	Pave	78	0.026721	1.000000

	LotShape	frequency	percentage	cumulative_perc
0	Reg	1859	0.636862	0.636862

## Matriz de Correlaciones

Una matriz de correlación es una tabla que indica los coeficientes de conexión entre los factores. Cada celda de la tabla muestra la conexión entre los dos factores (la matriz completa se encuentra en el código fuente de este trabajo). **Mas adelante en el trabajo se explica mas a detalle esta relación.**

1	0.0089	-0.028	-0.041	-0.03	-0.0028	-0.017	-0.05	-0.025	-0.017	0.018	-0.014	-0.025	-0.0087	-0.022	-0.038	-0.029	0.00015
0.0089	1	-0.42	-0.2	0.034	-0.066	0.034	0.043	0.0054	-0.064	-0.073	-0.13	-0.22	-0.25	0.31	0.026	0.072	0.00099
-0.028	-0.42	1	0.49	0.22	-0.076	0.12	0.092	0.22	0.22	0.047	0.11	0.35	0.46	0.027	0.0049	0.38	0.11
-0.041	-0.2	0.49	1	0.1	-0.036	0.024	0.022	0.13	0.19	0.084	0.021	0.25	0.33	0.032	0.00055	0.28	0.13
-0.03	0.034	0.22	0.1	1	-0.094	0.6	0.57	0.43	0.28	-0.043	0.28	0.55	0.48	0.25	-0.048	0.58	0.16
-0.0028	-0.066	-0.076	-0.036	-0.094	1	-0.37	0.048	-0.14	-0.05	0.042	-0.14	-0.17	-0.16	0.0055	0.009	-0.12	-0.042
-0.017	0.034	0.12	0.024	0.6	-0.37	1	0.61	0.31	0.28	-0.028	0.13	0.41	0.31	0.018	-0.14	0.24	0.21
-0.05	0.043	0.092	0.022	0.57	0.048	0.61	1	0.2	0.15	-0.062	0.17	0.3	0.24	0.16	-0.06	0.32	0.13
-0.025	0.0054	0.22	0.13	0.43	-0.14	0.31	0.2	1	0.3	-0.016	0.09	0.4	0.4	0.12	-0.058	0.4	0.14
-0.017	-0.064	0.22	0.19	0.28	-0.05	0.28	0.15	0.3	1	-0.055	-0.48	0.54	0.46	-0.16	-0.066	0.21	0.64
0.018	-0.073	0.047	0.084	-0.043	0.042	-0.028	-0.062	-0.016	-0.055	1	-0.24	0.089	0.084	-0.098	-0.0049	-0.018	0.16
-0.014	-0.13	0.11	0.021	0.28	-0.14	0.13	0.17	0.09	-0.48	-0.24	1	0.41	0.3	-0.00038	0.047	0.23	-0.4
-0.025	-0.22	0.35	0.25	0.55	-0.17	0.41	0.3	0.4	0.54	0.089	0.41	1	0.8	-0.21	-0.023	0.45	0.33
-0.0087	-0.25	0.46	0.33	0.48	-0.16	0.31	0.24	0.4	0.46	0.084	0.3	0.8	1	-0.25	-0.013	0.56	0.26
-0.022	0.31	0.027	0.032	0.25	0.0055	0.018	0.16	0.12	-0.16	-0.098	-0.00038	-0.21	-0.25	1	0.018	0.66	-0.16
-0.038	0.026	0.0049	0.00055	-0.048	0.009	-0.14	-0.06	-0.058	-0.066	-0.0049	0.047	-0.023	-0.013	0.018	1	0.097	-0.047
-0.029	0.072	0.38	0.28	0.58	-0.12	0.24	0.32	0.4	0.21	-0.018	0.23	0.45	0.56	0.66	0.097	1	0.061
0.00015	0.0099	0.11	0.13	0.16	-0.042	0.21	0.13	0.14	0.64	0.16	-0.4	0.33	0.26	-0.16	-0.047	0.061	1
0.01	-0.0019	-0.026	0.026	-0.041	0.084	-0.03	-0.046	0.015	0.078	0.099	-0.11	0.012	0.011	-0.06	-0.013	-0.044	-0.15
-0.0099	0.14	0.18	0.13	0.53	-0.22	0.47	0.46	0.26	0.082	-0.075	0.27	0.33	0.37	0.4	-0.0029	0.63	-0.019
-0.015	0.18	0.039	0.034	0.27	-0.089	0.27	0.21	0.19	-0.0073	-0.032	-0.036	-0.056	-0.1	0.61	-0.04	0.43	-0.033
0.0031	-0.0088	0.23	0.13	0.073	-0.0085	-0.053	-0.022	0.078	-0.11	-0.031	0.18	0.053	0.11	0.5	0.07	0.52	-0.16
-0.012	0.26	0.0047	-0.021	-0.16	-0.087	-0.14	-0.14	-0.051	-0.086	-0.038	0.065	-0.039	0.076	0.069	0.00044	0.12	-0.018
-0.029	0.041	0.35	0.21	0.39	-0.092	0.11	0.2	0.28	0.052	-0.048	0.25	0.28	0.39	0.58	0.1	0.81	-0.039
-0.035	-0.055	0.26	0.26	0.39	-0.031	0.17	0.13	0.28	0.29	0.066	0.0048	0.33	0.41	0.17	-0.0066	0.46	0.17
-0.027	0.088	0.077	-0.0086	0.57	-0.33	0.83	0.65	0.26	0.19	-0.069	0.17	0.35	0.26	0.086	-0.052	0.27	0.15
-0.01	-0.047	0.31	0.18	0.6	-0.18	0.54	0.43	0.36	0.26	-0.015	0.18	0.44	0.44	0.18	-0.067	0.49	0.16
-0.0089	-0.1	0.36	0.21	0.57	-0.15	0.48	0.38	0.37	0.31	0.0031	0.16	0.49	0.49	0.13	-0.054	0.49	0.18
-0.0071	-0.018	0.12	0.16	0.26	0.02	0.23	0.22	0.17	0.22	0.098	-0.039	0.23	0.23	0.09	-0.016	0.25	0.19
0.01	-0.016	0.16	0.1	0.3	-0.069	0.2	0.24	0.14	0.12	-0.0059	0.12	0.25	0.24	0.19	-0.00069	0.34	0.081
0.022	-0.021	0.012	0.021	-0.14	0.071	-0.37	-0.22	-0.11	-0.1	0.033	0.005	-0.086	-0.066	0.055	0.087	0.0033	-0.068
-0.047	-0.038	0.028	0.016	0.019	0.044	0.016	0.037	0.014	0.051	-0.023	-0.0058	0.038	0.044	-0.032	-0.045	0.0063	0.027
0.022	-0.049	0.076	0.054	0.043	0.044	-0.041	-0.047	0.065	0.097	0.063	-0.049	0.075	0.098	0.011	0.0068	0.086	0.053
0.014	-0.0031	0.17	0.094	0.031	-0.017	0.0023	-0.011	0.0045	0.084	0.045	-0.032	0.072	0.12	0.045	0.035	0.14	0.044
0.0082	-0.029	0.044	0.069	0.0056	0.034	-0.011	-0.0031	0.045	0.093	-0.0051	-0.01	0.084	0.093	-0.0053	-0.006	0.067	-0.0046
0.0064	-0.0012	0.011	0.0042	0.03	-0.0063	0.014	0.018	-0.00012	-0.00094	-0.0096	0.023	0.018	0.04	0.014	0.012	0.044	-0.0036
-0.26	-0.015	-0.0079	-0.024	-0.02	0.03	-0.012	0.033	-0.019	0.023	0.0089	-0.038	-0.011	-0.013	-0.019	-0.0023	-0.027	0.045
-0.0078	-0.059	0.26	0.24	0.55	-0.055	0.37	0.35	0.34	0.27	-0.0077	0.15	0.43	0.42	0.23	-0.019	0.52	0.16

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

## Encoder

En el proceso de procesamiento de datos, a veces necesitamos digitalizar números discretos o texto, Cuando se usa Python para el procesamiento de datos, es muy fácil usar el codificador para convertir variables ficticias (datos virtuales). El codificador puede convertir el texto en el conjunto de datos en un valor de 0 o 1. LabelEncoder y OneHotEncoder son dos funciones en el paquete scikit-learn, que pueden implementar el proceso de conversión anterior.

Se uso el encoder para así no borrar ningún dato de la base de datos, cuando aparecen datos categóricos en el conjunto de datos que queremos analizar, los datos en este momento no son ideales, **porque no podemos tratarlos matemáticamente**. Así de esta forma , no se borra ningún dato de la base de datos y se puede procesar, por que, aunque sean NaN o estén blanco, son datos a procesar que dan una mayor eficacia a nuestro modelo de **árbol y random forest**.

```
In [ ]: #Al aplicar el Encoder , Llenamos los datos faltantes, caracteres extraños o NaN que existan en los datos
le = preprocessing.LabelEncoder()
data = data.apply(le.fit_transform)
#data = data.apply(le.inverse_transform)
#Separamos los datos de test y train.
train,test = train_test_split(data,test_size=0.2)
```

Una vez que se entrenan nuestros datos , separamos nuestras variables dependientes e independientes.

```
In [ ]: #variables dependientes e independientes
train_x = train[['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond',
'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl',
'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

```
test_x = test[['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
              'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
              'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
              'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond',
              'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl',
              'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
              'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
              'BsmtExposure', 'BsmtFinType1', 'BsmtFinSE1', 'BsmtFinType2']
```

## Árbol de decisiones

Es un algoritmo de clasificación y regresión para su uso en el modelado predictivo de atributos discretos y continuos. Se crea primero el modelo y se entrena.

```
In [ ]: #Aplicamos Regresion con Arboles de desicion y entrenamos el modelo
modelo = DecisionTreeRegressor()
modelo.fit (train_x,train_y)
prediccion_arb = modelo.predict(test_x)
#Evaluacion del modelo
print("SVM" , metrics.accuracy_score(prediccion_arb,test_y))
print("MAE" , mean_absolute_error(test_y, prediccion_arb))
print("MSE" , mean_squared_error(test_y , prediccion_arb))
print("R2" , r2_score(test_y,prediccion_arb))
```

Se obtienen las primeras predicciones y primeros análisis del modelo construido:

**Error cuadrático medio, mean square error (MSE).** - se define como la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado

**Error absoluto medio, mean absolute error (MAE).** - se define como la diferencia en valor absoluto entre el valor real y el valor predicho.

**Raíz del error cuadrático medio, root mean square Error (RMSE).** - se define como la raíz cuadrada de la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado.

**R2.-** el coeficiente de determinación, determina la capacidad de un modelo para predecir futuros resultados



Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

```
SVM 0.5256849315068494
MAE 26.892123287671232
MSE 3081.707191780822
R2 0.7943841046703279
```

Como observamos en el análisis, nuestro modelo de **Árbol de Decisión** tiene un coeficiente de predicción de 0.794384, lo cual se puede considerar alto.

### Podado de árbol y estructura final

La estrategia para generar árboles más pequeños (poda) y con una menor varianza suele ser generar un árbol muy grande y después podarlo para obtener un sub-árbol. Así solo se enfoca en las variables importantes del modelo.

```
In [11]: #valores de ccp_alpha evaluados
param_grid = {"ccp_alpha": np.linspace(0,50,20)}
#busqueda por validacion cruzada
grid = GridSearchCV(
    #el arbol se crece al maximo posible para luego aplicar el pruning
    estimator = DecisionTreeRegressor(
        max_depth = None,
        min_samples_split=20,
        min_samples_leaf=1,
        random_state=123
    ),
    param_grid = param_grid,
    cv = 10,
    refit = True,
    return_train_score = True
)

grid.fit(train_x, train_y)
modelo_final = grid.best_estimator_
#print(f"Profundidad del arbol : {modelo_final.get_depth()}")
#print(f"Numero de nodos terminales : {modelo_final.get_n_leaves()}")
#estructura final
modelo_final = grid.best_estimator_
print(f"Profundidad del arbol : {modelo_final.get_depth()}")
print(f"Numero de nodos terminales : {modelo_final.get_n_leaves()}")
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	



Se vuelven a analizar las métricas del modelo, comparando con el modelo de **Árbol de Decisión sin podar**.

```
In [14]: predicción_final = modelo_final.predict(test_x)
print("MAE", mean_absolute_error(test_y, predicción_final))
print("MSE", mean_squared_error(test_y, predicción_final))
print("R2", r2_score(test_y, predicción_final))

MAE 28.373971566506814
MSE 3062.9074631743947
R2 0.795638449352936
```

SVM 0.5256849315068494  
MAE 26.892123287671232  
MSE 3081.707191780822  
R2 0.7943841046703279

MAE 28.373971566506814  
MSE 3062.9074631743947  
R2 0.795638449352936

## Conclusión:

Como se puede observar el modelo de **Árbol de Decisión con poda** tiene mejor predicción que el **Árbol de Decisión sin podar**

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

## Random Forest

Es un método versátil de aprendizaje automático capaz de realizar tanto tareas de regresión como de clasificación. También lleva a cabo métodos de reducción dimensional, trata valores perdidos, valores atípicos y otros pasos esenciales de exploración de datos. Es un tipo de método de aprendizaje por conjuntos, donde un grupo de modelos débiles se combinan para formar un modelo poderoso. Se puede decir que son varios árboles de decisión en un solo modelo.

### Creación del modelo y entrenamiento

```
In [15]: #Creacion del modelo Random Forest y entrenamiento
rf_model = RandomForestRegressor(
    n_estimators=25,
    criterion = "mse",
    max_depth = None,
    max_features = "auto",
    oob_score = False,
    n_jobs = -1 ,
    random_state = 123
)
rf_model.fit(train_x,train_y)
```

Se obtienen las primeras predicciones y primeros análisis del modelo construido, el R2 indica que **Random Forest** tiene una mayor predicción que los **Arboles de Decisión** para este ejemplo.

```
In [16]: #Evaluacion del modelo Random Forest

prediccion_rd = rf_model.predict(test_x)
#Evaluacion del modelo
#print("SVM" , metrics.accuracy_score(prediccion_rd,test_y))
print("MAE" , mean_absolute_error(test_y, prediccion_rd))
print("MSE" , mean_squared_error(test_y , prediccion_rd))
print("R2" , r2_score(test_y,prediccion_rd))
```



Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

MAE 20.58027397260274  
MSE 1891.859287671233  
R2 0.8737724523894539

MAE 28.373971566506814  
MSE 3062.9074631743947  
R2 0.795638449352936

## Lista de importancia de cada variable para la regresión de Random Forest y Árboles de Decisión.

Como se dijo anteriormente, la matriz de correlación y la lista de importancia tiene gran significado en el análisis de los datos ya que muestra las variables mas importantes y su relación con el entorno. En la siguiente grafica podemos ver un listado de mayor a menor las variables que se usan en el algoritmo para sus predicciones. (la matriz y el listado completos se encuentra en el código fuente de este trabajo).

	Caracteristicas	Importancia
0	Id	0.311383
17	OverallQual	0.096795
46	GrLivArea	0.095448
27	ExterQual	0.058689
19	YearBuilt	0.035174
...	...	...
71	PoolArea	0.000355
45	LowQualFinSF	0.000351
74	MiscFeature	0.000304
5	Street	0.000262
9	Utilities	0.000035
80 rows x 2 columns		

1	0.0009	-0.028	-0.041	-0.03	-0.0028	-0.017	-0.05	-0.025	-0.017	0.018	-0.014	-0.025	-0.0087	-0.022	-0.038	-0.029	0.00015
0.0009	1	-0.42	-0.2	0.034	-0.066	0.034	0.043	0.0054	-0.064	-0.073	-0.13	-0.22	-0.25	0.113	0.026	0.072	0.0099
-0.028	-0.42	1	0.49	0.22	-0.076	0.32	0.092	0.22	0.22	0.047	0.11	0.35	0.46	0.027	0.0049	0.38	0.11
-0.041	-0.2	0.49	1	0.1	-0.036	0.024	0.022	0.13	0.19	0.084	0.021	0.25	0.38	0.032	0.00055	0.28	0.13
0.03	0.034	0.22	0.1	1	-0.094	0.6	0.57	0.43	0.28	-0.043	0.28	0.55	0.48	0.15	-0.048	0.58	0.16
-0.0028	-0.066	-0.076	-0.036	-0.094	1	-0.37	0.048	-0.14	-0.05	0.042	-0.14	-0.17	-0.16	0.0055	0.009	-0.12	-0.042
-0.017	0.034	0.12	0.024	0.6	-0.37	1	0.51	0.31	0.23	-0.008	0.13	0.41	0.31	0.018	-0.14	-0.21	0.21
-0.05	0.043	0.092	0.022	0.57	0.048	0.61	1	0.7	0.35	0.062	0.17	0.3	0.24	0.16	-0.06	0.32	0.13
-0.025	0.0054	0.22	0.13	0.43	-0.14	0.31	0.2	1	0.1	-0.016	0.09	0.4	0.4	0.12	-0.058	0.4	0.14
-0.017	-0.064	0.22	0.19	0.28	-0.05	0.31	0.15	0.1	1	-0.055	-0.48	0.54	0.46	-0.16	-0.066	0.21	0.05
0.018	-0.073	0.047	0.084	-0.043	0.042	-0.028	-0.062	-0.016	-0.055	1	-0.24	0.089	0.084	-0.098	-0.0049	-0.018	0.16
-0.014	-0.13	0.11	0.021	0.28	-0.14	0.13	0.17	0.09	-0.48	-0.24	1	0.41	0.3	-0.00038	0.047	0.23	-0.4
-0.025	-0.22	0.15	0.22	0.55	-0.17	0.11	0.1	0.1	0.54	0.069	0.41	1	-0.8	-0.21	-0.023	0.45	-0.13
-0.0087	-0.25	0.46	0.33	0.48	-0.16	0.31	0.24	0.4	0.46	0.084	0.3	0.8	1	-0.25	-0.013	0.56	0.26
-0.022	0.31	0.027	0.032	-0.2	0.0055	0.018	0.16	0.12	-0.16	-0.098	-0.00038	-0.21	-0.25	1	0.018	0.66	-0.16
-0.038	0.026	0.0489	0.00055	-0.048	0.009	-0.14	-0.06	-0.058	-0.066	-0.0049	0.047	-0.023	0.013	0.018	1	0.007	-0.047
-0.029	0.072	0.38	0.28	0.58	-0.12	0.24	0.32	0.4	0.21	-0.018	0.23	0.45	0.56	0.66	0.097	1	0.061
0.00015	0.0099	0.11	0.13	0.16	-0.042	0.21	0.13	0.14	0.64	0.16	-0.4	0.33	0.24	-0.16	-0.047	0.061	1
0.01	-0.0019	-0.026	0.026	-0.041	0.084	-0.03	0.046	0.015	0.019	-0.11	0.012	0.011	-0.06	-0.013	-0.044	-0.15	0.0015
0.0099	0.14	0.18	0.13	0.53	-0.22	0.47	0.46	0.26	0.082	0.075	0.27	0.33	0.37	0.4	-0.0029	0.63	-0.019
-0.015	0.14	0.039	0.034	0.27	-0.089	0.27	0.21	0.19	-0.0973	-0.032	-0.036	-0.006	-0.1	0.61	-0.04	0.49	-0.033
0.0013	-0.0008	0.13	0.13	0.073	-0.0085	0.053	0.022	0.078	-0.11	-0.031	0.18	0.053	0.11	0.51	0.07	0.52	-0.16
-0.012	-0.26	0.0047	-0.021	-0.16	-0.087	-0.14	-0.14	-0.051	-0.086	-0.038	0.065	-0.039	0.076	0.009	0.00044	0.12	-0.018
-0.029	0.041	0.25	0.21	0.27	-0.092	0.11	0.2	0.27	0.052	0.048	0.25	0.21	0.27	0.55	0.1	0.61	-0.039
-0.035	-0.055	0.26	0.26	0.39	-0.031	0.17	0.13	0.28	0.29	0.066	0.0048	0.33	0.41	0.17	-0.0066	0.46	0.17
-0.027	0.088	0.077	-0.0086	0.57	-0.33	0.83	0.65	0.26	0.19	0.069	0.17	0.35	0.26	0.086	-0.052	0.27	0.15
-0.01	-0.047	0.17	0.18	0.4	-0.18	0.54	0.48	0.16	0.74	-0.015	0.18	0.44	0.44	0.18	-0.067	0.49	0.16
-0.0089	-0.1	0.16	0.21	0.57	-0.15	0.48	0.38	0.37	0.31	0.0031	0.16	0.49	0.49	0.13	-0.054	0.49	0.18
-0.0071	0.018	0.12	0.16	0.56	0.02	0.23	0.22	0.17	0.22	0.008	-0.039	0.23	0.21	0.09	-0.016	0.26	0.19
0.01	-0.016	0.16	0.1	0.3	-0.049	0.2	0.21	0.14	0.32	0.0019	0.12	0.35	0.24	0.19	-0.00049	0.34	0.081
0.022	-0.021	0.012	0.021	-0.14	0.071	0.37	0.22	-0.11	-0.1	0.033	0.005	-0.086	-0.066	0.055	0.087	0.0033	-0.068
-0.047	0.038	0.028	0.016	0.019	0.044	0.016	0.037	0.014	0.051	-0.023	-0.0058	0.038	0.044	-0.032	-0.0045	0.0063	0.027
0.022	-0.049	0.076	0.054	0.043	0.044	-0.041	-0.047	0.065	0.097	0.063	-0.049	0.075	0.098	0.011	-0.0068	0.006	0.053
0.014	-0.0031	0.17	0.094	0.031	-0.017	0.0023	-0.011	0.0045	0.084	0.045	-0.032	0.072	0.12	0.045	0.035	0.14	0.044
0.0082	-0.029	0.044	0.069	0.0056	0.034	-0.011	-0.0031	0.045	0.093	-0.0051	-0.01	0.084	0.093	-0.0053	-0.006	0.067	-0.0046
0.0064	-0.0012	0.011	0.0042	0.03	-0.0063	0.014	0.018	-0.00012	-0.00004	-0.0096	0.023	0.018	0.04	0.014	0.012	0.044	-0.0036
-0.26	-0.015	-0.0079	-0.024	-0.02	0.03	-0.012	0.033	-0.019	0.023	0.0089	-0.038	-0.011	-0.013	-0.019	-0.0023	0.027	0.045
-0.0078	-0.059	0.26	0.28	0.55	-0.055	0.37	0.35	0.34	0.27	-0.0077	0.15	0.43	0.42	-0.23	-0.019	0.52	0.16

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

**Comenta las ventajas y desventajas de cada modelo. De acuerdo con los resultados, ¿son realmente útiles los modelos creados para el conjunto de datos propuesto?**

El modelo de **Arboles de Decisión** resulto tener un poco menos de confiabilidad que **Random Forest**, esto debido a la gran cantidad de datos que el algoritmo tiene que usar. Al momento de usar **Random Forest** y clasificarlos en diferentes arboles (y no en uno solo como lo hace el algoritmo de **Arboles de Decisión**), se puede obtener más fiabilidad.

### Algoritmos de Clasificación

Los algoritmos de clasificación nos permiten, como su nombre lo dice, ordenar información de una manera especial basándonos en un criterio de ordenamiento. Se usan cuando el resultado deseado es una etiqueta discreta, en otras palabras, son útiles cuando la respuesta al problema cae dentro de un conjunto finito de resultados posibles.

### Discretizando los datos

Se empieza a cortar a hacer los grupos con respecto a los precios de las casas

```
#Se empieza a cortar a hacer los grupos con respecto a los precios de las casas

data_c["Clasificacion"] = pd.cut(data_c["SalePrice"], bins=[34900, 100000, 500000, 755000], labels=["Grupo1",
```

Comprobamos que realmente se hizo la clasificación:

```
data_c["Clasificacion"].value_counts()

Grupo2    2787
Grupo1     122
Grupo3       9
Name: Clasificacion, dtype: int64
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

Se divide el conjunto de datos en (X) variables de características y (Y) variable objeto

```
In [ ]: #dividimos el conjunto de datos en (X) variables de características y (Y) variable objeto
data_c = data_c.apply(1e.fit_transform)
X = data_c[["SalePrice" , "Clasificacion"]]
Y =data_c["Clasificacion"]
```

## Modelo de clasificación

Se empieza por hacer el modelo de clasificación por árboles, se crea el modelo y se entrena, además obtenemos nuestros primeros análisis del modelo

```
In [ ]: x_train,x_test , y_train , y_test = train_test_split(X,Y ,train_size = 0.5 , random_state = 0)
#entrenar el modelo
modelo_arb_c = DecisionTreeClassifier(random_state = 1)
modelo_arb_c.fit(x_train,y_train)
predicciones = modelo_arb_c.predict(X = x_test,)
print(classification_report(y_test , predicciones))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	63
1	1.00	1.00	1.00	1395
2	1.00	1.00	1.00	2
accuracy			1.00	1460
macro avg	0.99	1.00	1.00	1460
weighted avg	1.00	1.00	1.00	1460

Como se puede observar en el análisis, tenemos un Accuracy y precisión muy altos , del 98% y 99% respectivamente, lo que indica que nuestro modelo puede predecir el grupo sin ningún problema.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

## Cross-Validation K-Fold

```
#Cross-Validation
kf = KFold(n_splits = 5)#numero de iteraciones
puntuaciones = cross_val_score(modelo_arb_c,x_train , y_train , cv = kf , scoring = "accuracy")
print ("Metricas Cross_validation" , puntuaciones)

Metricas Cross_validation [0.99657534 1.          0.99657534 1.          1.          ]
```

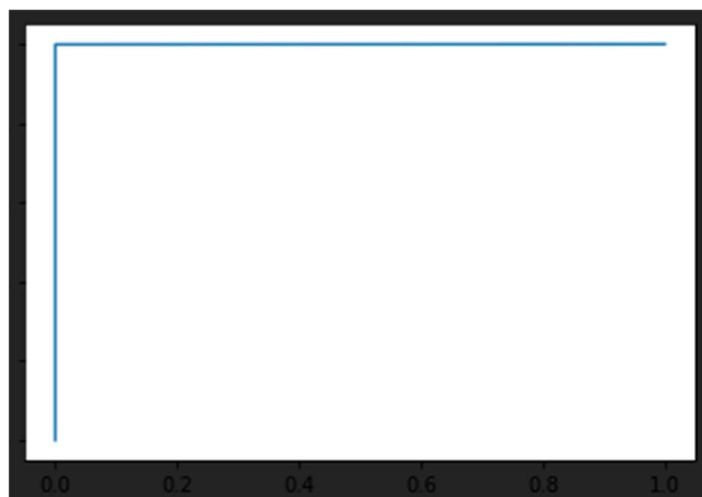
## Accuracy

```
#Accuracy
accuracy = accuracy_score( y_true = y_test , y_pred = predicciones, normalize = True)
print(f"el accuracy de test es : {100*accuracy} %")

el accuracy de test es : 99.93150684931507 %
```

## Curvas ROC/AUC

cuando se trata de un problema de clasificación, podemos contar con una curva AUC-ROC. Esta es una de las métricas de evaluación más importante para verificar el rendimiento de cualquier modelo de clasificación. Esta métrica se utiliza para determinar el balance entre la detección de verdaderos positivos y evitar los falsos positivos. Para ello, se muestra la proporción de detección de los verdaderos positivos en el eje vertical y la proporción de los falsos positivos en el eje horizontal, para un umbral o punto de corte determinado



Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	21/03/2022
	Nombre: Aldo Alberto	

## Conclusiones

Para los algoritmos de regresión usados en este ejercicio , Arboles de Decisión y Random Forest, se obtuvo un rendimiento y una predicción alta, considerando la cantidad de datos usados. Muchos de estos datos se podrían considerar innecesarios (espacios en blanco, caracteres raros o NaN) pero para el algoritmo son necesarios por que tiene una **mas fiabilidad** al momento de determinar el precio de venta de la casa. Para ambos algoritmos se obtuvo una predicción muy alta , para el modelo usado.

En el caso de los algoritmos de regresión, se obtuvo resultados y análisis de predicción muy altos del casi .99%, lo cual indica que el modelo creado es perfecto para el uso de predicciones en categorías para la base de datos.

## Bibliografías:

\*Análisis de Datos categóricos con Python. (s. f.). Python. Recuperado 21 de marzo de 2022, de <https://relopezbriega.github.io/blog/2016/02/29/analisis-de-datos-categoricos-con-python/>

\*Método de preprocesamiento de datos para digitalizar datos de categoría- LabelEncoder VS OneHotEncoder - programador clic. (s. f.). Python. Recuperado 21 de marzo de 2022, de <https://programmerclick.com/article/77681608357/>

\**funpymodeling*. (2020, 16 septiembre). PyPI.

<https://pypi.org/project/funpymodeling/>