WALTER RUBIO

MAGDALENA VILLAN

MARCOS SAYONI

Objetivo

- Estimar un modelo de regresión lineal que realice predicciones para el precio por metro cuadrado.
- Usar cross-validation para validar el modelo. Deberá prestar cierta atención a la estructura espacial de los precios.
- Aplicar regularización a modelos lineales pueden hacerlo para obtener un puntaje adicional.
- Seleccionar un portafolio (de manera aleatoria) de 100 propiedades. Determinar cuáles de las propiedades se encuentran sobrevaluados o subvaluados y en qué magnitud.
- Teniendo en cuenta que podría comprar y vender propiedades al precio de mercado, omitamos costos de transacción, con un capital inicial igual al valor de mercado de las propiedades en su portafolio. ¿Cuál es el mejor portafolio de propiedades que podría comprar?

NORMALIZAMOS

X con Min-Max

Normalizamos los X con MinMax o Standard Scaler

```
|: from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import scale

#Uso MinMax o StandardScaler
#minmaxscaler_model = StandardScaler()
minmaxscaler_model = MinMaxScaler()

# Columnas a normalizar
columnas= [CAMPO_SUPERFICIE]

# Si calcula superficie por ambiente y uso ROOMS
if CALCULAR_SUPERFICIE_POR_AMBIENTE & UTILIZAR_ROOMS:
    # También normalizar rooms por m2
    columnas.append('m2_por_habitacion')

elif UTILIZAR_ROOMS & (~CALCULAR_SUPERFICIE_POR_AMBIENTE):
    columnas.append('habitaciones')

# Aplico scaler a las columnas elegidas
modelo_df[columnas] = minmaxscaler_model.fit_transform(modelo_df[columnas])
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:21: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

DUMMIES

TIPO PROPIEDAD
UBICACIÓN

Tipo de propiedad

Obtenemos las dummies de **tipo_propiedad**

```
# Obtenemos las columnas dummies de property type
df_tipo_propiedad = pd.get_dummies(modelo_df['tipo_propiedad'], drop_first=True, prefix='tipo_propiedad')

# Agregamos las columnas dummies concatenando el dataframe original con las dummies
df_with_dummies = pd.concat([modelo_df, df_tipo_propiedad], axis=1)

# Eliminamos la columna tipo_propiedad del dataframe nuevo
df_with_dummies = df_with_dummies.drop('tipo_propiedad', axis=1)
```

Ubicación

Obtenemos las dummies de **ubicacion** .

```
# Obtenemos la columna dummies de ubicacion
df_ubicacion = pd.get_dummies(modelo_df['ubicacion'], drop_first=True, prefix='ubicacion')

# Agreamos las columnas dummies concatenando el nuevo dataframe con las dummies
df_with_dummies = pd.concat([df_with_dummies, df_ubicacion], axis=1)

# Eliminamos la columna place_with_parent_names del nuevo dataframe
df_with_dummies = df_with_dummies.drop('ubicacion', axis=1)

# Guardamos el nombre de cada ubicacion en la variable ubicacion
ubicacion = df_ubicacion.columns
```

```
#guardo en una columna el valor de los indices para no perderlos
df_with_dummies['indices']=df_with_dummies.index
```

PREPARAMOS DATOS PARA ENTRENAMIENTO

Definimos Valores y Labels

Como vamos a entrenar el modelo

Creamos DF donde vamos imputando los resultados

Cross-Validation

Valores y Labels Del dataframe, separamos nuestros labels de los valores a utilizar para entrenar

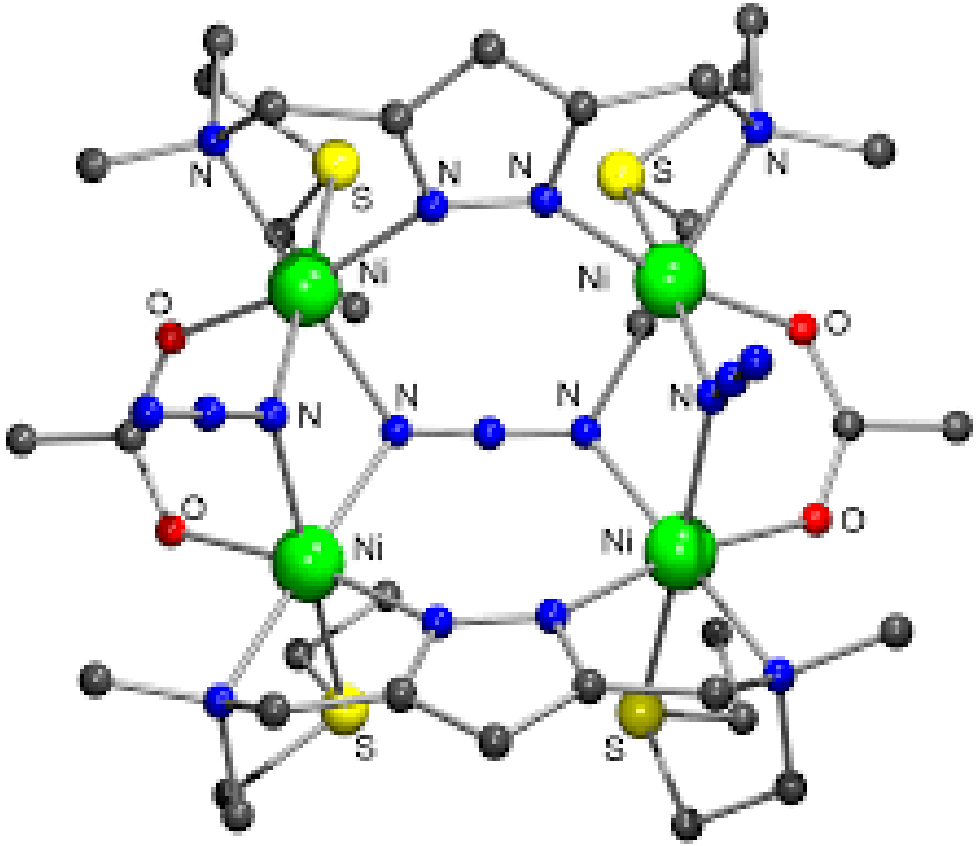
```
def get_features_labels(df):
    #Separa las features de los labels de un dataframe y los devuelve en el orden FEATURES, LABELS (X, Y)
    if PRECIO:
        # Obtengo los labels (precio)
        Y = df['precio']

        # Obtengo los valores sin los labels
        X = df.drop('precio', axis=1)
    else:
        # Obtengo los labels (precio)
        Y = df['precio_usd_por_m2']

        # Obtengo los valores sin los labels
        X = df.drop('precio_usd_por_m2', axis=1)


    return X, Y
```

Holdout Sets

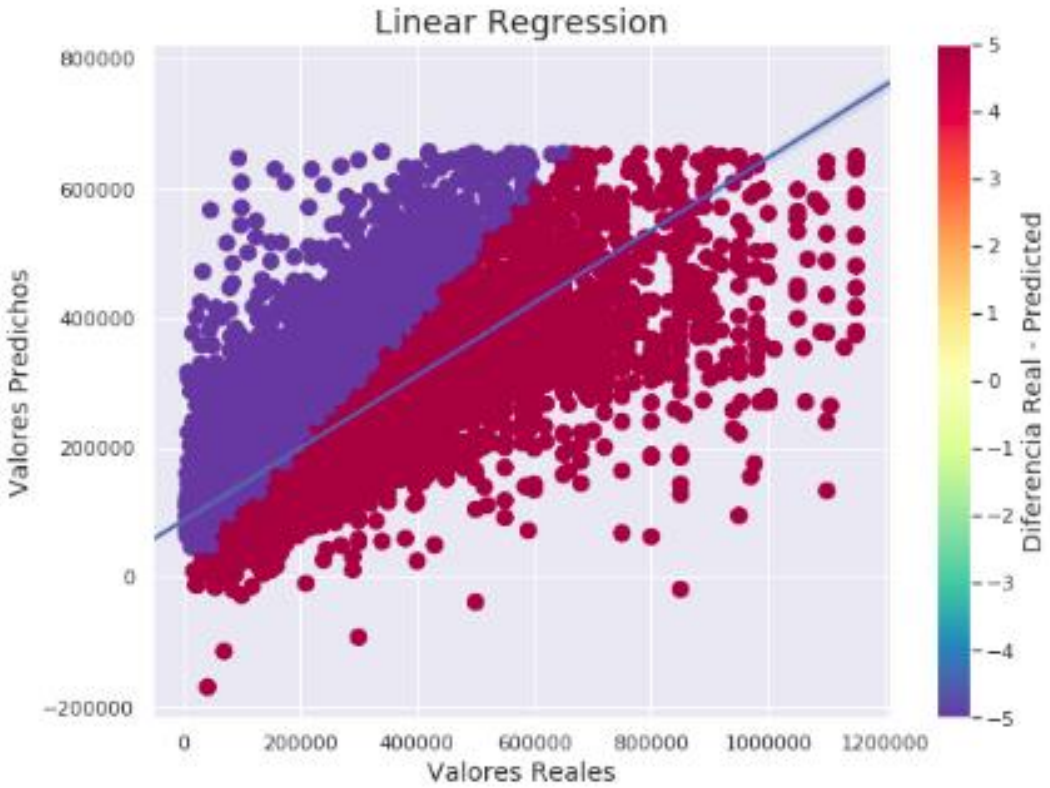
De nuestros datos de entrenamiento con labels conocidos, separamos un porcentaje con el cual entrenaremos el modelo, y otro porcentaje con el cual haremos la validación. Vamos a usar un 70% para el entrenamiento y un 30% para la validación.

```
# Utilizamos el método train_test_split de Scikit Learn para separar los datos
from sklearn.model_selection import train_test_split

def get_houldout_sets(X,Y, random_state, train_size):

    # obtenemos X e Y de test y entrenamiento.
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=random_state, train_size=train_size)

    return X_train, X_test, Y_train, Y_test
```

APLICAMOS DIFERENTES MODELOS

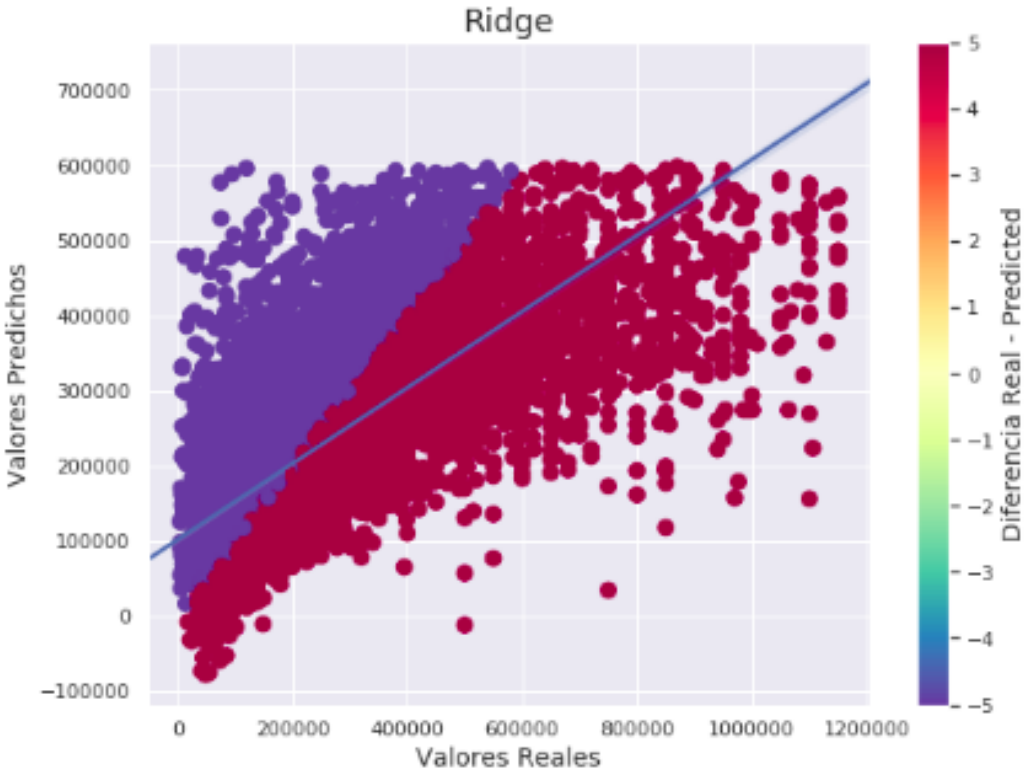


MODELO de REGRESION LINEAL



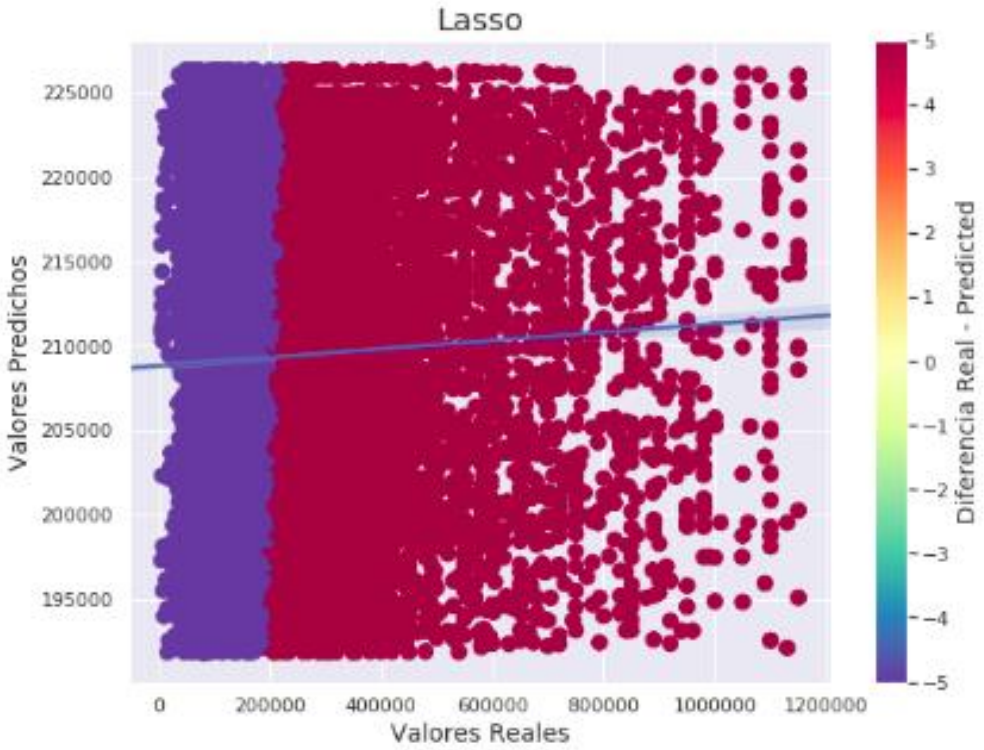
	Model	MSE	R^2	Alpha	Cant. Cols.
0	Linear Regression	1.600986e+10	0.489016	0	720.0

MODELO de REGRESION RIDGE



	Model	MSE	R^2	Alpha	Cant. Cols.
0	Linear Regression	1.600986e+10	0.489016	0	720.0
1	Ridge	1.427282e+10	0.560067	1	720.0

MODELO de REGRESION LASSO



	Model	MSE	R^2	Alpha	Cant. Cols.
0	Linear Regression	1.600986e+10	0.489016	0	720.0
1	Ridge	1.427282e+10	0.560067	1	720.0
2	Lasso	3.155169e+10	0.002627	315039	720.0

PORTAFOLIO de PROPIEDADES

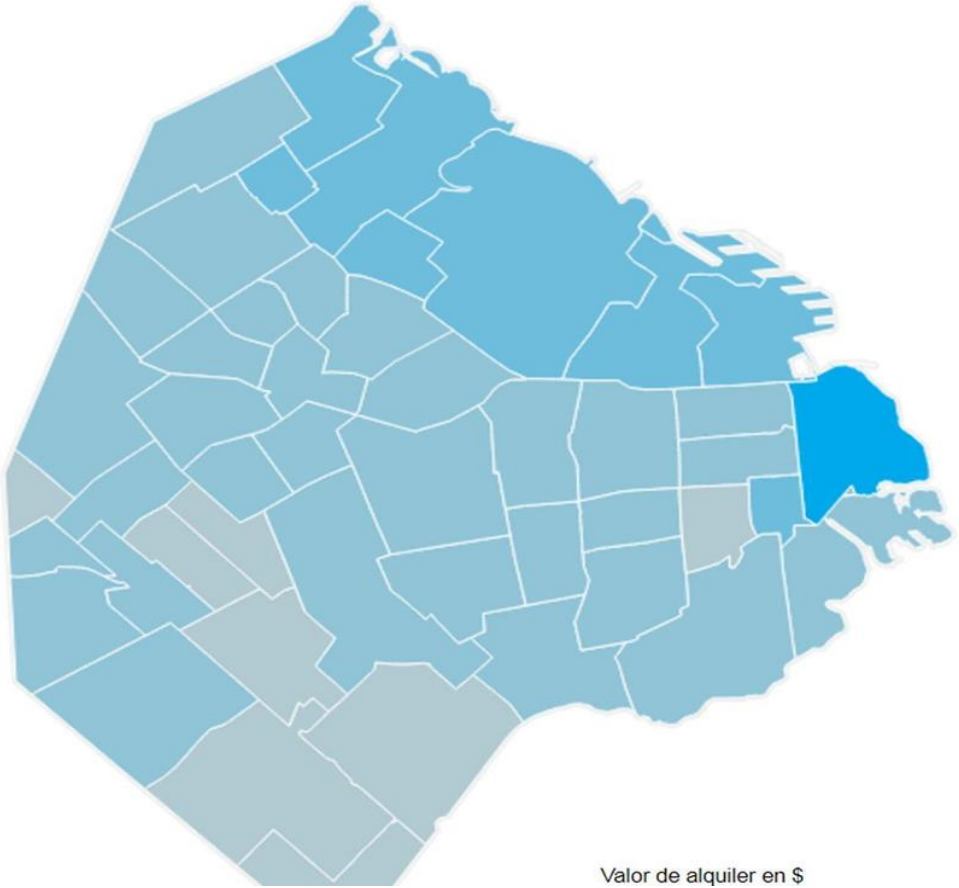
Modelo Ridge con el mejor

R2 0.56

Tomamos una muestra de 100 valores del dataframe original por un total de valor de $ 22.7718.201 (precio de venta)

Vemos propiedades en las que el modelo tuvo una diferencia de +- $ 1.000.

	precio	prediccion	ubicacion	tipo_propiedad	diferencia
45638	48792.790	49790.139	Santa Fe-Rosario	apartment	-997.349
63561	117500.000	118490.633	Capital Federal-Villa Urquiza	apartment	-990.633
56418	44306.100	45295.376	Santa Fe-Rosario	apartment	-989.276
56376	44306.100	45290.543	Santa Fe-Rosario	apartment	-984.443
43704	102215.020	103199.064	Capital Federal-Villa Luro	apartment	-984.044
16095	76000.000	76981.999	Capital Federal-Almagro	apartment	-981.999
3801	298000.000	298976.258	Bs.As. G.B.A. Zona Norte-Tigre	house	-976.258
3800	298000.000	298976.179	Bs.As. G.B.A. Zona Norte-Tigre	house	-976.179
4550	480000.000	480974.637	Bs.As. G.B.A. Zona Norte-San Isidro	house	-974.637



Valor de alquiler en $

1	Puerto Madero	18.605
2	Palermo	11.310
3	Núñez	10.570
4	Colegiales	10.496
5	Belgrano	9.919
6	Recoleta	9.803
7	Retiro	9.492
8	Coghlan	9.364
9	San Telmo	9.269
10	Villa Urquiza	8.844
11	Saavedra	8.831
12	Chacarita	8.801
13	Villa Crespo	8.316
14	Villa Ortúzar	8.283
15	Almagro	8.185
16	Parque Chas	8.184
17	Monserrat	8.151
18	Caballito	8.145
19	Villa Pueyrredón	8.106
20	San Nicolás	8.023
21	Agronomía	7.954
22	Versalles	7.910
23	Pque. Chacabuco	7.887
24	Villa Devoto	7.809
25	Pque. Patricios	7.766
26	Barracas	7.710