

Código de Regresión Lineal

Manuel Montufar Galnares

Junio 2024

El presente trabajo es para hacer una regresión lineal con un frame work, para este trabajo usare sklearn, específicamente el modelo Gradient Boosting Regressor, ya que es el que yo considero como un gran regresor.

La parte de transformación de datos la ahorraré en este espacio, ya que es la misma que utilice para la actividad pasada, por lo que los resultados del apartado de transformación en el código es el mismo que el de la primera actividad

```
"""
Importar Librerías
"""

import pandas as pd # Lectura y Transformación de Datos
import matplotlib.pyplot as plt # Visualización de Datos
import seaborn as sns
import numpy as np # Manejo de arreglos y operaciones de vectores

from sklearn.model_selection import cross_val_score,
    train_test_split, KFold, learning_curve, GridSearchCV
from sklearn.metrics import make_scorer, mean_squared_error,
    mean_absolute_error, r2_score, explained_variance_score
from sklearn.ensemble import GradientBoostingRegressor
```

Estas serán las librerías que usaremos para el modelo

```
"""
Definir el target
"""
# Variables predictoras (X) y objetivo (y)
X = df[['year', 'mmr']]
y = df[['sellingprice']]

# Transformarlo a un array unidimensional
y = y.values.ravel()
```

Aquí definimos la variable target y nos aseguramos que este en una dimensión

```
"""
Validación Cruzada
"""

# Definir el modelo
gbr = GradientBoostingRegressor(random_state=42)

# Definir las métricas para la validación cruzada
scoring = {
    'MSE': make_scorer(mean_squared_error, greater_is_better=False),
    'MAE': make_scorer(mean_absolute_error, greater_is_better=False),
    'R2': make_scorer(r2_score),
    'RMSE': make_scorer(lambda y, y_pred:
        np.sqrt(mean_squared_error(y, y_pred)), greater_is_better=False),
    'Explained Variance': make_scorer(explained_variance_score)
}
```

```
# Validacion cruzada
cv = KFold(n_splits=5, shuffle=True, random_state=42)
results = {}
for metric_name, metric in scoring.items():
    scores = cross_val_score(gbr, X, y, cv=cv, scoring=metric)
    results[metric_name] = scores.mean(), scores.std()

# Mostrar resultados de validacion cruzada
results_df = pd.DataFrame(results, index=['Mean', 'Std'])
print(results_df)
```

Aquí hacemos la validacion cruzada del modelo y obtenemos lo siguiente:

	MSE	MAE	R2	RMSE	Explained Variance
Mean	-3.090444e+06	-1095.342744	0.967498	-1757.366919	0.967498
Std	1.634958e+05	3.681566	0.001436	45.887042	0.001436

Figura 1: Resultados de la validación cruzada

```
"""
Entrenamiento del
Modelo
"""

# Dividir los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Asegurate de que y_train y y_test sean vectores unidimensionales
y_train = y_train.ravel()
y_test = y_test.ravel()

# Entrenar el modelo solo con los datos de entrenamiento
gbr.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = gbr.predict(X_test)

# Calcular metricas utilizando los valores de prueba
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
ev = explained_variance_score(y_test, y_pred)
rmse = np.sqrt(mse)

# Mostrar metricas
metrics = {
    'MSE': mse,
    'MAE': mae,
    'RMSE': rmse,
    'R2': r2,
    'Explained_Variance': ev
}

metrics_df = pd.DataFrame(metrics, index=[0])
print(metrics_df)
```

Aquí entrenamos todo el modelo con el 80 % de los datos y 20 % para validación Y obtenemos los siguientes resultados:

	MSE	MAE	RMSE	R2	Explained Variance
0	2.940092e+06	1088.449448	1714.669669	0.968622	0.968622

Figura 2: Resultados del Modelo

```
"""
Curva de Aprendizaje
"""

# Obtener la curva de aprendizaje
train_sizes, train_scores, test_scores = learning_curve(
    gbr, X_train, y_train, cv=5, scoring='neg_mean_squared_error',
    train_sizes=np.linspace(0.1, 1.0, 10), n_jobs=-1
)

# Convertir los puntajes negativos de MSE a positivos
train_scores_mean = -np.mean(train_scores, axis=1)
test_scores_mean = -np.mean(test_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

# Visualizar la curva de aprendizaje
plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_scores_mean, 'o-', color='r',
         label='Puntuacion de Entrenamiento')
plt.plot(train_sizes, test_scores_mean, 'o-', color='g',
         label='Puntuacion de Validacion')
plt.fill_between(train_sizes, train_scores_mean
                 - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1, color='r')
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1, color='g')
plt.xlabel('Tamano del Conjunto de Entrenamiento')
plt.ylabel('Error Cuadrático Medio')
plt.title('Curva de Aprendizaje')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```

Aquí vemos la curva de aprendizaje del modelo para evaluar que no haya overfitting

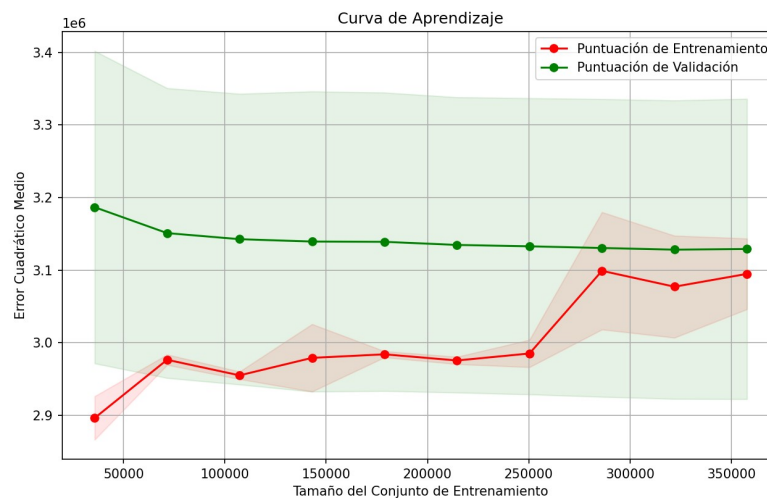


Figura 3: Curva de Aprendizaje del Modelo

```
"""
Grafico de Prediccion
"""

# Verificamos el tipo de dato
print(type(y_test))
print(type(y_pred))
print(y_test[:5])
print(y_pred[:5])

# Transformamos el tipo de dato
y_test_values = y_test.astype(float)

# Asegurate de que y_test_values y y_pred sean numpy arrays de tipo float
y_test_values = np.array(y_test_values, dtype=float)
y_pred = np.array(y_pred, dtype=float)

# Encontrar el valor minimo y maximo
min_val = min(np.min(y_test_values), np.min(y_pred))
max_val = max(np.max(y_test_values), np.max(y_pred))

# Crear el grafico
plt.figure(figsize=(10, 6))
plt.plot(y_test_values, y_pred, 'o', label='Predicciones vs Valor Real')
plt.plot([min_val, max_val], [min_val, max_val],
         color='red', linestyle='--', label='Linea de Igualdad')
plt.xlabel('Valor Real')
plt.ylabel('Prediccion')
plt.title('Predicciones vs Valor Real')
plt.legend()
plt.grid(True)
plt.show()
```

Finalmente graficamos la linea de predicciones:

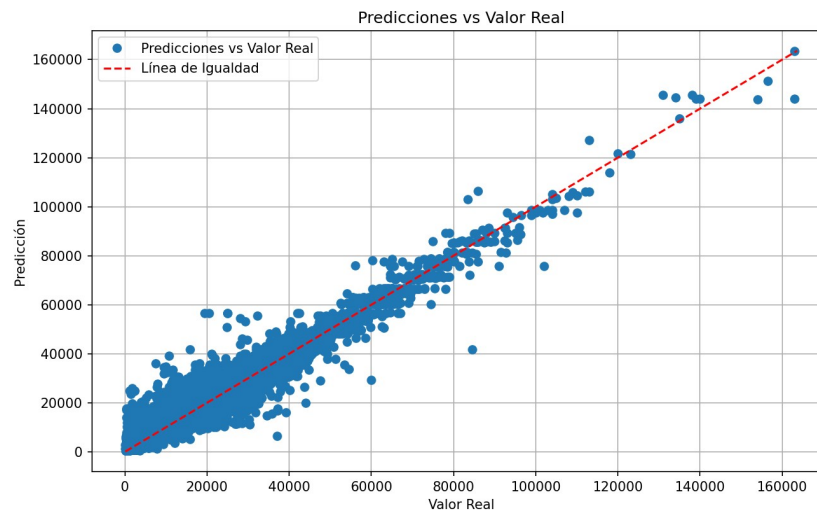


Figura 4: Predicciones del Modelo

Conclusión

Hacer un modelo en un frame work es mucho más rapido, más amigable y nos permite jugar con más características, a la vez tenemos más herramientas para medir que tan bueno es el modelo, tambien tenemos muchos más regresores con los que probar.

Ha sido una gran actividad para tener en mente la importancia que es entender que hacen los modelos por detras de la maquina, pero tambien para apreciar el tiempo que nos ahorra utilizar estos modelos desde los frame works.