

**“Predicción de tiempos de avance de obra, en el rubro de pintura en estructura metálica usando modelos de Machine Learning”**

**Christian Jose Bermeo Tenesaca**

**<https://www.linkedin.com/in/bermeo-christian>**

**Git-Hub [cro-ss \(Christian Bermeo\)](#)**

**Agosto 2025**

## **Antecedentes**

La pintura estructural es el proceso de aplicación de pintura mediante lo que se conoce como maquinas pulverizadoras de pintura (airless) sobre una superficie, en este caso sobre estructuras metálicas (vigas, columnas, conexiones).

## **Objetivo**

Teniendo en cuenta este marco de trabajo, donde las variables de las están relativamente contraladas. Cuando me refiero a variables quiero decir que sabes sobre que superficies estamos trabajando, con que máquinas estamos aplicando la pintura y que materiales (pinturas) estamos aplicando. Por lo tanto, podemos definir nuestro objetivo sobre este marco.

*El objetivo es pronosticar, predecir el tiempo que me tomará pintar nuevas superficies similares(vigas, columnas) con similares materiales y con las mismas máquinas dentro del mismo proyecto o en proyectos diferentes pero con las mismas condiciones de trabajo.*

*Ejemplo:*

*Si se que una primera etapa de un proyecto de pintura estructural pinté 5000m<sup>2</sup> de estructura metálica en 15 días, cuanto me tomará pintar 6000 m<sup>2</sup> en otro proyecto en similares condiciones.*

El problema, aunque parezca sencillo superficialmente depende de más variables como:

- Niveles de la estructura
- Diseño estructural(Dimesiones de vigas y columnas)
- Calidad de material
- Condiciones operativas
- Condiciones de personal
- Condiciones de las máquinas

Por lo que se busca resolver es dar un nivel más detallado del rendimiento del trabajo, predecir los tiempos de trabajo más acercados a la realidad y teniendo en cuenta estás variables. Incluso se puede llegar a predecir a los cuantos m<sup>2</sup> de aplicación de pintura las máquinas llegarán a fallar. O cada

cuantos m2 baja el rendimiento del personal. Y así como esas hay muchas preguntas más que se derivan de este problema.

## **Resumen**

### **Objetivo General**

- Predecir tiempo de aplicación de pintura metálica en proyectos similares con condiciones similares (rendimiento)

### **Objetivos Específicos**

- Predecir demoras, retrasos, fallas en la aplicación de pintura
- Predecir reparaciones en las máquinas pulverizadoras de pintura.
- Incrementar la cantidad de m2 pintados por plazos de tiempo
- Incrementar las ganancias de la compañía

## **Stakeholders**

**Inmobiliaria:** El encargado de desarrollar el proyecto y entregar la información necesaria como: planos, ordenes de trabajo, materiales, datos al contratista

**Contratista:** El encargado de aplicar la pintura y llevar el control y la predicción de sus rendimientos.

## **Documentación sobre los datos**

Los datos sobre las características de la estructura como:

- Dimesiones de vigas y columnas
- Ubicación o zonas a pintar dentro del proyecto
- Longitudes
- Cantidad de elementos a pintar
- Planos

Son entregados por la inmobiliaria, y de estos datos depende la estrategia del contratista para alcanzar la mayor productividad y rendimiento económico. Ya que con esta línea base de información puede llevar el control del avance de obra.

Ítem	Ubicación	Paño	Ejes	L	N	almas pintadas	h efec alma(mm)	patin inferior pintado	b patin(mm)	patin interior pintado	b efect patin interior (mm)	Total Area	Unidad
VP-14 I1100×350×15×20	Nivel +12.00 - Bloque 4	1	34(B-C),33(B-C)	3.94	2	1	1060	1	350	1	335	13.75	m2
VP-8 I750×220×10×15	Nivel +12.00 - Bloque 4	1	33-34(C )	12.25	1	2	720	1	220	2	210	25.48	m2
VP-8 I750×220×10×15	Nivel +12.00 - Bloque 4	1	33(C-D)	9.30	1	2	720	1	220	2	210	19.34	m2
VP-8 I750×220×10×15	Nivel +12.00 - Bloque 4	1	34(C-D)	9.30	1	1	720	1	220	1	210	10.7	m2
VP-1 I500×200×8×12	Nivel +12.00 - Bloque 4	1	33-34(C-D)/(1)	1.90	1	2	476	1	200	2	192	2.92	m2
VP-1 I500×200×8×12	Nivel +12.00 - Bloque 4	1	(C-D)/(33-34)/(2)	3.90	1	2	476	1	200	2	192	5.99	m2
B-6 I750×200×8×12	Nivel +12.00 - Bloque 4	1	33-34(B-C)	13.00	2.5	2	726	1	200	2	192	66.17	m2

**Tabla 1: Baseline**

La segunda parte es la más importante y es el control de obra, que es realizado principalmente por el contratista. Aquí se lleva el control del tiempo que lleva pintar los elementos y superficies.

Información creada por el contratista

- Cantidad Ejecutada
- Día en el que se aplicó la pintura
- Que material se aplicó en determinado día

Realizado 1/0	Cantidad Ejecutada Cuacho Clorado	Fecha de Progreso Cuacho Clorado	Realizado 1/0	Cantidad Ejecutada Intumescente	Fecha de Progreso Intumescente
1	13.75	<b>martes, 10 de junio de 2025</b>	1	13.75	lunes, 23 de junio de 2025
1	25.48	<b>martes, 10 de junio de 2025</b>	1	25.48	lunes, 23 de junio de 2025
1	19.34	<b>martes, 10 de junio de 2025</b>	1	19.34	lunes, 23 de junio de 2025
1	10.7	<b>martes, 10 de junio de 2025</b>	1	10.7	lunes, 23 de junio de 2025
1	2.92	<b>martes, 10 de junio de 2025</b>	1	2.92	lunes, 23 de junio de 2025
1	5.99	<b>martes, 10 de junio de 2025</b>	1	5.99	lunes, 23 de junio de 2025
1	66.17	<b>martes, 10 de junio de 2025</b>	1	66.17	lunes, 23 de junio de 2025

-

## Metodología del despliegue

La propuesta es sencilla, estoy buscando una predicción de tiempos de avance de pintura. Por lo tanto, voy a escoger una regresión lineal o un random forest para predecir mis valores y los voy a comparar y evaluar. Estamos prediciendo variables continuas entonces nuestro marco de trabajo se reduce a estos algoritmos que cumplen con lo que necesitamos.

## Análisis y preprocesamiento de datos.

### EDA

En este caso no voy a rellenar valores faltantes, quiero trabajar con datos completos y reales.

- Corregimos valores nulos.
- Corregimos variables discretas que deberían ser continuas (“L,” h efecto alma” )

```
paint_df.info()
```

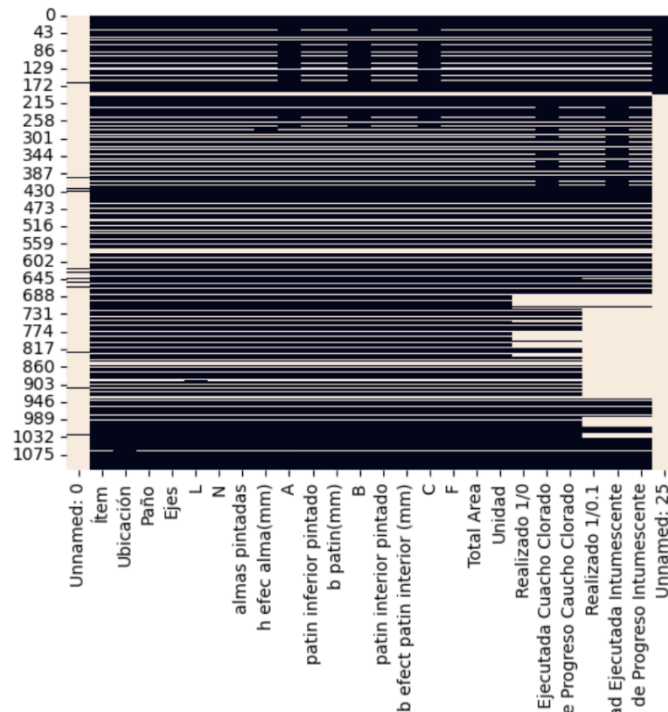
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1113 entries, 0 to 1112
Data columns (total 26 columns):
 #   Column                                     Non-Null Count  Dtype
---  -
 0   Unnamed: 0                               44 non-null     object
 1   Ítem                                     874 non-null     object
 2   Ubicación                               883 non-null     object
 3   Paño                                     874 non-null     float64
 4   Ejes                                     874 non-null     object
 5   L                                         875 non-null     object
 6   N                                         874 non-null     float64
 7   almas pintadas                          872 non-null     float64
 8   h efec alma(mm)                        873 non-null     object
 9   A                                         906 non-null     float64
10   patin inferior pintado                  872 non-null     float64
11   b patin(mm)                            872 non-null     float64
12   B                                         907 non-null     float64
13   patin interior pintado                  872 non-null     float64
14   b efect patin interior (mm)             872 non-null     float64
15   C                                         908 non-null     float64
16   F                                         872 non-null     float64
17   Total Area                             874 non-null     float64
18   Unidad                                  874 non-null     object
19   Realizado 1/0                           815 non-null     float64
20   Cantidad Ejecutada Cuacho Clorado       837 non-null     float64
21   Fecha de Progreso Caucho Clorado        815 non-null     datetime64[ns]
22   Realizado 1/0.1                         659 non-null     float64
23   Cantidad Ejecutada Intumescente         690 non-null     float64
24   Fecha de Progreso Intumescente          659 non-null     datetime64[ns]
25   Unnamed: 25                             193 non-null     object
dtypes: datetime64[ns](2), float64(16), object(8)
memory usage: 226.2+ KB
```

## Mapa de calor

Representativo para entender como esta mi df. En este caso veo muchas filas vacías, porque de esa manera crearon el dataset.

```
[55]: sns.heatmap(paint_df.isnull(),cbar=False)
```

```
[55]: <Axes: >
```



## Vistazo rápido a los datos

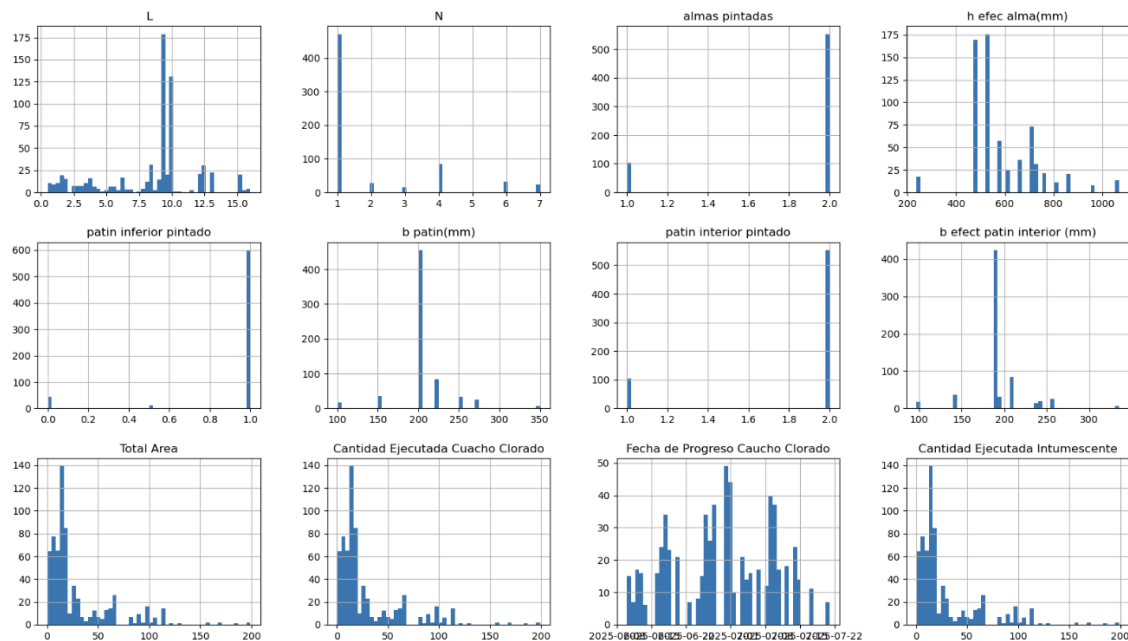
No vemos ninguna anomalía entre los datos, como outliers, duplicados

```
115]: paint_df['L'] = pd.to_numeric(paint_df['L'], errors='coerce')
paint_df['h efec alma(mm)'] = pd.to_numeric(paint_df['h efec alma(mm)'], errors='coerce')
paint_df.describe()
```

115]:

	L	N	almas pintadas	h efec alma(mm)	patin inferior pintado	b patin(mm)	patin interior pintado	b efect patin interior (mm)	Total Area	Cantidad Ejecutada Caucho Colorado	Fecha de Progreso Caucho Colorado	Cantidad Ejecutada Intumesciente	Fec
count	657.000000	657.000000	657.000000	657.000000	657.000000	657.000000	657.000000	657.000000	657.000000	657.000000	657	657.000000	
mean	8.502938	1.942922	1.843227	592.621005	0.922374	203.881279	1.841705	195.423135	28.793181	28.793181	2025-06-29 06:21:22.191780864	28.793181	17.4
min	0.500000	1.000000	1.000000	238.000000	0.000000	100.000000	1.000000	97.000000	0.500000	0.500000	2025-06-10 00:00:00	0.500000	
25%	7.900000	1.000000	2.000000	484.000000	1.000000	200.000000	2.000000	192.000000	8.980000	8.980000	2025-06-20 00:00:00	8.980000	
50%	9.300000	1.000000	2.000000	530.000000	1.000000	200.000000	2.000000	192.000000	16.060000	16.060000	2025-06-30 00:00:00	16.060000	
75%	10.000000	2.000000	2.000000	720.000000	1.000000	200.000000	2.000000	194.000000	31.200000	31.200000	2025-07-09 00:00:00	31.200000	
max	16.000000	7.000000	2.000000	1070.000000	1.000000	350.000000	2.000000	335.000000	198.930000	198.930000	2025-07-21 00:00:00	198.930000	
std	3.467391	1.720620	0.363864	144.804045	0.259106	30.955385	0.365296	29.507656	32.316961	32.316961	NaN	32.316961	

```
[129]: ## Some graphic information
paint_df.hist(bins=50, figsize=(20,15))
plt.show()
```



```
paint_df.duplicated(keep=False).value_counts()
```

```
: False      657
Name: count, dtype: int64
```

## Feature Engineering and Feature Selection

### Codificación y Normalización

Primero vamos a realizar una codificación de las variables discretas, para este caso vamos a utilizar label encoder en vez del one hot encoder por la dimensionalidad que me genera. (Lo usamos a través del column transformer)

- Tengo 7 tipos de ubicaciones
- Tengo 21 tipos de elementos

Por ende es mejor no usar el one-hot encoding

```
>>]: from sklearn.preprocessing import LabelEncoder
import numpy as np

labels = paint_df['Ubicación']
encoder = LabelEncoder()
encoded_labels = encoder.fit_transform(labels)
labels_1 = paint_df['Ítem']
encoder_1 = LabelEncoder()
encoded_labels_1 = encoder_1.fit_transform(labels_1)
labels_2 = paint_df['Ejes']
encoder_2 = LabelEncoder()
encoded_labels_2 = encoder_2.fit_transform(labels_2)
print(np.argsort(np.unique(encoded_labels)))
print(np.argsort(np.unique(encoded_labels_1)))
```

```
[0 1 2 3 4 5 6 7]
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21]
```

La transformación esencial aquí es date time, yo no puedo trabajar con fechas sino con duraciones.

Aquí me di cuenta de un problema grande es que no puedo calcular ni pronosticar cuánto me tomará pintar nuevas vigas porque no tengo una



hora de inicio ni una hora de final si no que solo tengo una fecha global de cuando hice la actividad. Ese problema lo abordare en las conclusiones.

Por ahora voy a pronosticar cuanto tiempo me toma acabar una tarea y empezar otra.

Para este ejemplo seria cuanto tiempo me toma pintar caucho clorado para empezar el intumescente en los mismo elementos.

```
[212]: paint_df_model['diff_days'] = (paint_df_model['Fecha de Progreso Intumescente'] - paint_df_model['Fecha de Progreso Caucho Clorado']).dt.days
paint_df_model
```

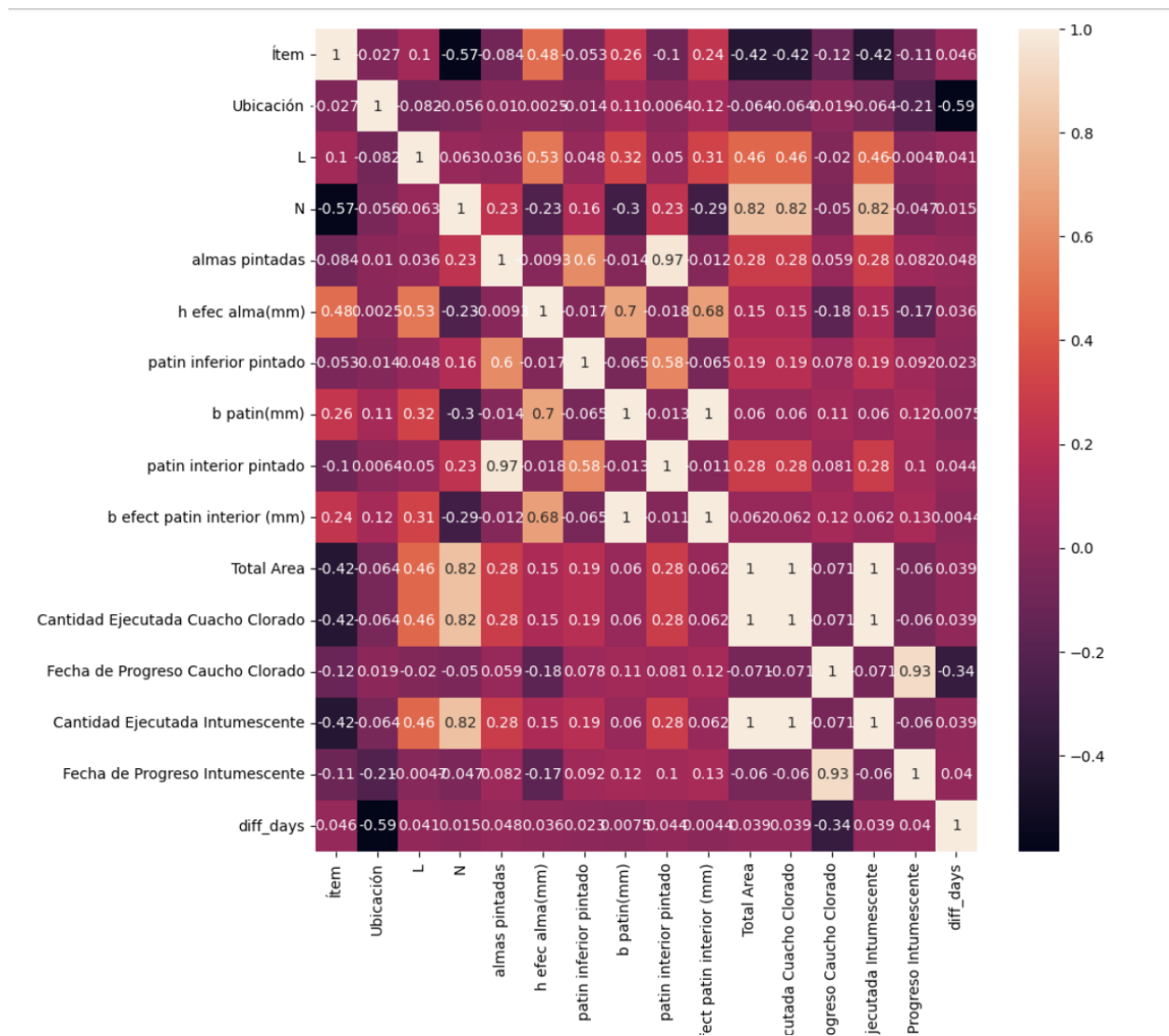
```
[212]:
```

	Ítem	Ubicación	L	N	almas pintadas	h efec alma(mm)	patin inferior pintado	b patin(mm)	patin interior pintado	b efect patin interior (mm)	Total Area	Cantidad Ejecutada Caucho Clorado	Fecha de Progreso Caucho Clorado	Cantidad Ejecutada Intumescente	Fecha de Progreso Intumescente	diff_days
0	10	3	3.94	2.0	1.0	1060	1.0	350.0	1.0	335.0	13.75	13.75	2025-06-10	13.75	2025-06-23	13
1	20	3	12.25	1.0	2.0	720	1.0	220.0	2.0	210.0	25.48	25.48	2025-06-10	25.48	2025-06-23	13
2	20	3	9.30	1.0	2.0	720	1.0	220.0	2.0	210.0	19.34	19.34	2025-06-10	19.34	2025-06-23	13
3	20	3	9.30	1.0	1.0	720	1.0	220.0	1.0	210.0	10.70	10.70	2025-06-10	10.70	2025-06-23	13
4	6	3	1.90	1.0	2.0	476	1.0	200.0	2.0	192.0	2.92	2.92	2025-06-10	2.92	2025-06-23	13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1108	20	6	6.30	1.0	2.0	720	1.0	220.0	2.0	210.0	13.10	13.10	2025-07-14	13.10	2025-07-21	7
1109	20	6	6.30	1.0	2.0	720	1.0	220.0	2.0	210.0	13.10	13.10	2025-07-14	13.10	2025-07-16	2
1110	6	6	6.70	1.0	2.0	476	1.0	200.0	2.0	192.0	10.29	10.29	2025-07-14	10.29	2025-07-22	8
1111	0	6	9.25	4.0	2.0	484	1.0	200.0	2.0	194.0	57.57	57.57	2025-07-15	57.57	2025-07-16	1
1112	0	6	1.21	4.0	2.0	484	1.0	200.0	2.0	194.0	7.53	7.53	2025-07-14	7.53	2025-07-22	8

657 rows x 16 columns

## Correlación

Elimino las características que tengan una gran correlación antes de eliminar feature y luego hacer feature engineering.



Mi dataset queda del siguiente tamaño considerando las características que son relevante

```
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Ítem                   657 non-null   int32
1   Ubicación              657 non-null   int32
2   L                      657 non-null   float64
3   N                      657 non-null   float64
4   almas pintadas         657 non-null   float64
5   h efec alma(mm)        657 non-null   int64
6   patin interior pintado 657 non-null   float64
7   Total Area             657 non-null   float64
8   diff_days              657 non-null   int64
dtypes: float64(5), int32(2), int64(2)
memory usage: 46.2 KB
```

## PRIMEROS RESULTADOS

Sin aplicar mucho feature engineering , vemos que existe un error muy grande al predecir los valores.

- Mean absolute error: 2,3153
- Mean squared error : 8,67
- R2: -0,78

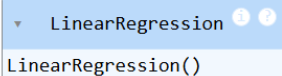
Para mí las dos métricas más importantes son R2 Y Mean absolute error

R2 porque me permite saber si los resultados que obtengo son mejores que los de azar pero necesito también del P-value.

Y Mean absolute error porque es simple de interpretar a menor valor mayor seguridad en las predicciones correctas

```
[317]: X_train, X_valid, y_train, y_valid = train_test_split(X,y, test_size=0.2, random_state=1)
```

```
[326]: lr= LinearRegression()  
modelo_1_trained= lr.fit(X_train, y_train)  
modelo_1_trained
```

```
[326]:  LinearRegression()
```

```
[340]: ## Importamos las metricas de error que vamos a evaluar.  
from sklearn.metrics import mean_absolute_error  
from sklearn.metrics import mean_squared_error  
from sklearn.metrics import r2_score # R2  
from sklearn.metrics import confusion_matrix # R2  
from sklearn.metrics import classification_report
```

```
[341]: ## Predecimos los valores  
y_pred= lr.predict(X_valid)  
pred_1= mean_absolute_error(y_pred,y_valid)  
pred_1
```

```
[341]: 2.315312558703732
```

```
[342]: pred_2= mean_squared_error(y_pred,y_valid)  
pred_2
```

```
[342]: 8.67480105131724
```

```
[343]: pred_3= r2_score(y_pred,y_valid)  
pred_3
```

```
[343]: -0.7837589029261838
```

**Conclusión 1 este modelo no es el más optimo**

## Modelo 2: Random Forest Classifier

Tambien genera demasiado error, no estamos obteniendo los resultados necesarios.

```
[347]: y=modelo_2.iloc[:,[-1]]
X=modelo_2.iloc[:,[0,1,2,3,4,5,6,7]]
X_train, X_valid, y_train, y_valid = train_test_split(X,y, test_size=0.2, random_state=1)
rfc= RandomForestClassifier(n_estimators= 10)
modelo_2_trained= rfc.fit(X_train, y_train)
modelo_2_trained

C:\Users\chris\anaconda3\envs\tensor_f\lib\site-packages\sklearn\base.py:1473: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using ravel().
    return fit_method(estimator, *args, **kwargs)

[347]: + RandomForestClassifier
RandomForestClassifier(n_estimators=10)

[349]: y_pred= rfc.predict(X_valid)
pred_1= mean_absolute_error(y_pred,y_valid)
pred_2= mean_squared_error(y_pred,y_valid)
pred_3= r2_score(y_pred,y_valid)
print(pred_1,pred_2,pred_3)

1.878787878787879 9.454545454545455 0.23785201737706296
```

## Usando la normalización

Tampoco obtenemos los mejores resultados.

```
1.9393939393939394 9.560606060606060 0.3266694421652122
```

```
[51]: y=modelo_3.iloc[:,[-1]]
      X=modelo_3.iloc[:,[0,1,2,3,4,5,6,7]]
      X_train, X_valid, y_train, y_valid = train_test_split(X,y, test_size=0.2, random_state=1)
```

```
## Normalization and Feature Engineering
# I will use and STANDAR SCALER AND SELECT KBEST
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
sc= StandardScaler()
X_train= sc.fit_transform(X_train)
X_valid= sc.transform(X_valid)
#y_train= sc.fit(y_train)
print([[X_valid]])
filtro= SelectKBest(score_func=f_regression, k=2)
X_filtrado = filtro.fit_transform(X,y)
print(X_filtrado)
```

```
[[array([[ 0.94425315, -1.38801524,  0.24504999, ..., -0.10666626,
          0.4145781 , -0.39978635],
        [-0.48602066,  0.84591486,  0.24504999, ..., -0.79109285,
          0.4145781 , -0.45778814],
        [-0.48602066,  0.84591486, -0.53158633, ..., -0.79109285,
          0.4145781 , -0.58582402],
        ...,
        [-0.48602066, -1.38801524,  0.20190353, ..., -0.79109285,
          0.4145781 , -0.65554959],
        [ 0.56609049,  0.22507422,  0.00000000, ...,  0.00000000,
          0.01859808  0.25444672])
```

```
[514]: r= LinearRegression()
      modelo_3_trained= lr.fit(X_train, y_train)
      modelo_3_trained
```

```
[514]: ▼ LinearRegression ⓘ ⓘ
      LinearRegression()
```

```
[515]: y_pred= lr.predict(X_valid)
      pred_3= mean_absolute_error(y_pred,y_valid)
      pred_2= mean_squared_error(y_pred,y_valid)
      pred_3= r2_score(y_pred,y_valid)
      print(pred_1,pred_2,pred_3)

1.9848484848484849 8.674801051317242 -0.7837589029261827
```

```
[ ]:
```

## Utilizando Select Kbest

### Tampoco vemos una mejora en los resultados de la regresión lineal

```
C:\Users\chris\anaconda3\envs\tensor_f\lib\site-packages\sklearn\base.py:1473: DataConversionWarning:
ctor y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for exam
avel().
    return fit_method(estimator, *args, **kwargs)
[0.09865645 0.22497003 0.21933953 0.06830324 0.01646999 0.09921596
 0.01859808 0.25444672]

580]: y=modelo_4.iloc[:,[-1]]
      x=modelo_4.iloc[:,[1,2,7]]
      X_train, X_valid, y_train, y_valid = train_test_split(X,y, test_size=0.2, random_state=1)

581]: ## Normalization and Feature Engineering
      # I will use and STANDAR SCALER AND SELECT KBEST
      from sklearn.preprocessing import StandardScaler
      from sklearn.feature_selection import SelectKBest, f_regression
      sc= StandardScaler()
      X_train= sc.fit_transform(X_train)
      X_valid= sc.transform(X_valid)
      filtro= SelectKBest(score_func=f_regression, k=2)
      X_filtrado = filtro.fit_transform(X,y)
      print(X_filtrado)

[[ 3.    3.94]
 [ 3.   12.25]
 [ 3.    9.3 ]
 ...
 [ 6.    6.7 ]
 [ 6.    9.25]
 [ 6.    1.21]]

C:\Users\chris\anaconda3\envs\tensor_f\lib\site-packages\sklearn\utils\validation.py:1339: DataConver:
A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
```

```
32]: LinearRegression
LinearRegression()
```

```
33]: y_pred= lr.predict(X_valid)
pred_3= mean_absolute_error(y_pred,y_valid)
pred_2= mean_squared_error(y_pred,y_valid)
pred_3= r2_score(y_pred,y_valid)
print(pred_1,pred_2,pred_3)

1.8106060606060606 8.814757826812373 -0.9214218059254886
```

```
34]:
```

## Conclusiones

- Al ir modelando mi información me di cuenta de una falla en la recolección de datos, lo que implico tener buscar otras características a predecir
- El resultado que estoy buscando talvez no tenga ninguna relación con los datos(Estos buscando los días que hay entre una actividad y otra entre elementos pintados). Para este caso los días que pasan para pintar la misma viga con caucho clorado e intumesciente.
- El proyecto no tuvo pipelines ni despliegue a producción, debido a los horarios no pude tomar esas clases pero queda pendiente modificar el dataset y estudiar los diferentes maneras de enviarlo a producción.