Lo primero que hicimos fue importar las librerías que usamos en el código:

```
import numpy as np
import pandas as pd
pd.set_option('max_columns', None)
pd.set_option('max_rows', 90)

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')

!pip install pycaret
from sklearn.neighbors import KNeighborsRegressor
import scipy.stats
from sklearn.preprocessing import StandardScaler
from pycaret.regression import setup, compare_models
from sklearn.model_selection import KFold, cross_val_score
```

Ahora importamos los dataset de la competencia y almacenamos su contenido en las siguientes variables:

```
from google.colab import files

train0 = pd.read_csv('train.csv')
test0 = pd.read_csv('test.csv')
sample_submission = pd.read_csv('sample_submission.csv')
```

El reto es pronosticar el precio de venta de las casas, por lo tanto, nuestra variable objetivo será la columna “SalePrice” perteneciente a el dataset train0. Eliminamos entonces del data set las columnas “Id” y “SalePrice”, luego concatenamos las columnas restantes y las guardamos en una variable distinta:

```
target = train0['SalePrice']
test_ids = test0['Id']

train1 = train0.drop(['Id', 'SalePrice'], axis=1)
test1 = test0.drop('Id', axis=1)

data1 = pd.concat([train1, test1], axis=0).reset_index(drop=True)
data1
```

Ahora procedemos a limpiar los datos, empezamos por los datos que no son promediables, es decir los No Numéricos. Para ello creamos una copia de la concatenación anterior:

```
data2 = data1.copy()

data2['MSSubClass'] = data2['MSSubClass'].astype(str)

for column in ['Alley','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','Bsmt
               'GarageQual','GarageCond','PoolQC','Fence','MiscFeature']:
    data2[column] = data2[column].fillna("None")

for column in ['MSZoning','Utilities','Exterior1st','Exterior2nd','MasVnrType','
    data2[column] = data2[column].fillna(data2[column].mode()[0])
```

Continuamos con los valores numéricos vacíos, para esto creamos una función que nos permite rellenar estos valores, esta función que almacenará en la variable Y_train los valores de na_target que no son vacíos de cada correspondiente columna del dataset mediante una comparación que nos arroja las posiciones de los valores que no son vacíos (==False) y que son vacíos (==True). Los valores que no son vacíos del resto del dataset se almacenarán en la variable X_train. Finalmente en la variable X_test se almacenan los valores de na_target que contienen valores numéricos vacíos. Después con Regressor hacemos un ajuste para luego hacer una predicción de los valores perdidos y finalmente reemplazamos en donde se encuentran los valores vacíos con los valores predecidos anteriormente.

```
def value_knn(df, na_target):
    df = df.copy()

    numeric_df = df.select_dtypes(np.number)
    non_na_columns = numeric_df.loc[: ,numeric_df.isna().sum() == 0].columns

    y_train = numeric_df.loc[numeric_df[na_target].isna() == False, na_target]
    X_train = numeric_df.loc[numeric_df[na_target].isna() == False, non_na_columns]
    X_test = numeric_df.loc[numeric_df[na_target].isna() == True, non_na_columns]

    knn = KNeighborsRegressor()
    knn.fit(X_train, y_train)

    y_pred = knn.predict(X_test)

    df.loc[df[na_target].isna() == True, na_target] = y_pred

    return df
```

De esta manera limpiamos de datos vacíos nuestro dataset:

```
for column in ['LotFrontage','MasVnrArea','BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','BsmtFullBath','BsmtHalfBath','GarageYrBlt',
               'GarageCars','GarageArea']:
    data3 = value_knn(data3, column)
```

Y hasta aquí vamos de momento…