PROYECTO SBERBANK RUSSIAN HOUSING MARKET - MODELOS Y SIMULACION
ENTREGA FINAL

Programa:
INGENIERÍA DE SISTEMAS

Por:
JOHAN SEBASTIAN HENAO CAÑAS - 1000085432

Docente:
RAUL RAMOS POLLAN

MEDELLIN

2023

Consideración:

Para el curso en general se presentaron varios inconvenientes que me llevaron a cambiar el proyecto en dos ocasiones. Inicialmente como aun no se necesitaban los datos para la primera entrega elegí una competición que su dataset tenia un peso de 16 GB, trate de manejarlo borrándole datos, pero finalmente opte por cambiar el proyecto.

Por ultimo para la entrega numero dos elegí otra competición y tenia un reto muy interesante de predecir 206 valores por cada registro, hice el procesado de los datos, que de hecho no era mucho lo que se tenia que procesar, pero fue un reto que se me complico, de nuevo por la alta dimensionalidad y porque era un problema de clasificación múltiple, tenía muchas dimensiones tanto en filas como en columnas y cada que corría un código se tardaba mucho tiempo; ni con las técnicas de PCA pude disminuir el tiempo.

Ahora opte por desarrollar la presente competencia.

Introducción:

En el dinámico paisaje inmobiliario, entender las complejidades del mercado es esencial para tomar decisiones informadas y estratégicas. Este informe busca proporcionar una visión profunda del mercado inmobiliario, centrándose en la predicción de precios de viviendas en función de una variedad de variables clave. Al explorar las relaciones entre factores económicos, características de la propiedad y dinámicas de vecindario, pretendemos ofrecer a los inversores y propietarios una herramienta valiosa para maximizar sus decisiones financieras.

En el contexto actual, marcado por fluctuaciones económicas y cambios en las preferencias del consumidor, la estabilidad del mercado inmobiliario adquiere una relevancia crítica. Examinaremos detalladamente las tendencias recientes y los indicadores macroeconómicos relevantes para proporcionar una base sólida para nuestras proyecciones. Desde el PIB hasta tasas de interés y el comportamiento de los mercados de capitales, cada aspecto será meticulosamente analizado para brindar una panorámica completa.

El núcleo de nuestro análisis se basa en un conjunto de datos exhaustivo que abarca diversas dimensiones. Desde detalles específicos de la propiedad, como área total y condiciones de construcción, hasta características demográficas y la presencia de servicios en los alrededores, cada elemento del conjunto de datos contribuye a la formación de un panorama detallado. Este enfoque integral permitirá no solo comprender la dinámica del mercado, sino también anticipar futuros cambios.

El valor de una propiedad no solo radica en sus características internas, sino también en su entorno. Nuestro análisis se extiende a las características del vecindario, incluyendo la

proximidad a servicios públicos, instalaciones recreativas y la calidad ambiental. Este enfoque holístico reconoce la interconexión de factores que influyen en las decisiones de compra, proporcionando una perspectiva más completa para evaluar el valor de una propiedad.

El contexto macroeconómico, delineado en nuestro informe mediante datos clave, desempeña un papel crucial en la predicción de los precios inmobiliarios. Desde tasas de intercambio hasta el crecimiento del PIB y la estabilidad financiera, entender cómo estos factores afectan el mercado local es esencial. Este informe se sumergirá en estas variables para proporcionar una visión clara de la interrelación entre el entorno económico más amplio y el mercado inmobiliario local.

Con el objetivo de brindar claridad y utilidad práctica, este informe se enfoca en alcanzar varios objetivos clave. Desde el análisis detallado de variables hasta la construcción de modelos predictivos, cada sección está diseñada para proporcionar información valiosa. A través de esta estructura, aspiramos a empoderar a nuestros lectores con conocimientos sólidos y perspicacia estratégica para navegar el complejo panorama del mercado inmobiliario.

Información del dataset:

Este conjunto de datos tiene como objetivo principal predecir el precio de venta de cada propiedad, utilizando la variable objetivo, llamada "price_doc" en el archivo train.csv. El conjunto de entrenamiento abarca desde agosto de 2011 hasta junio de 2015, mientras que el conjunto de prueba comprende el período de julio de 2015 a mayo de 2016. Además de la información detallada sobre transacciones individuales, el conjunto incluye datos sobre las condiciones generales de la economía y el sector financiero de Rusia. Esto permite enfocarse en generar pronósticos precisos de precios para propiedades individuales sin tener que anticipar el ciclo económico.

train.csv, test.csv: Contienen información sobre transacciones individuales, con filas indexadas por el campo "id", que se refiere a transacciones individuales (algunas propiedades pueden aparecer más de una vez en transacciones separadas). También incluyen información complementaria sobre el área local de cada propiedad.

El conjunto de entrenamiento (train.csv) consta de 30,471 entradas y 292 columnas, con datos de agosto de 2011 a junio de 2015.

El conjunto de prueba (test.csv) tiene 7,662 entradas y 291 columnas.

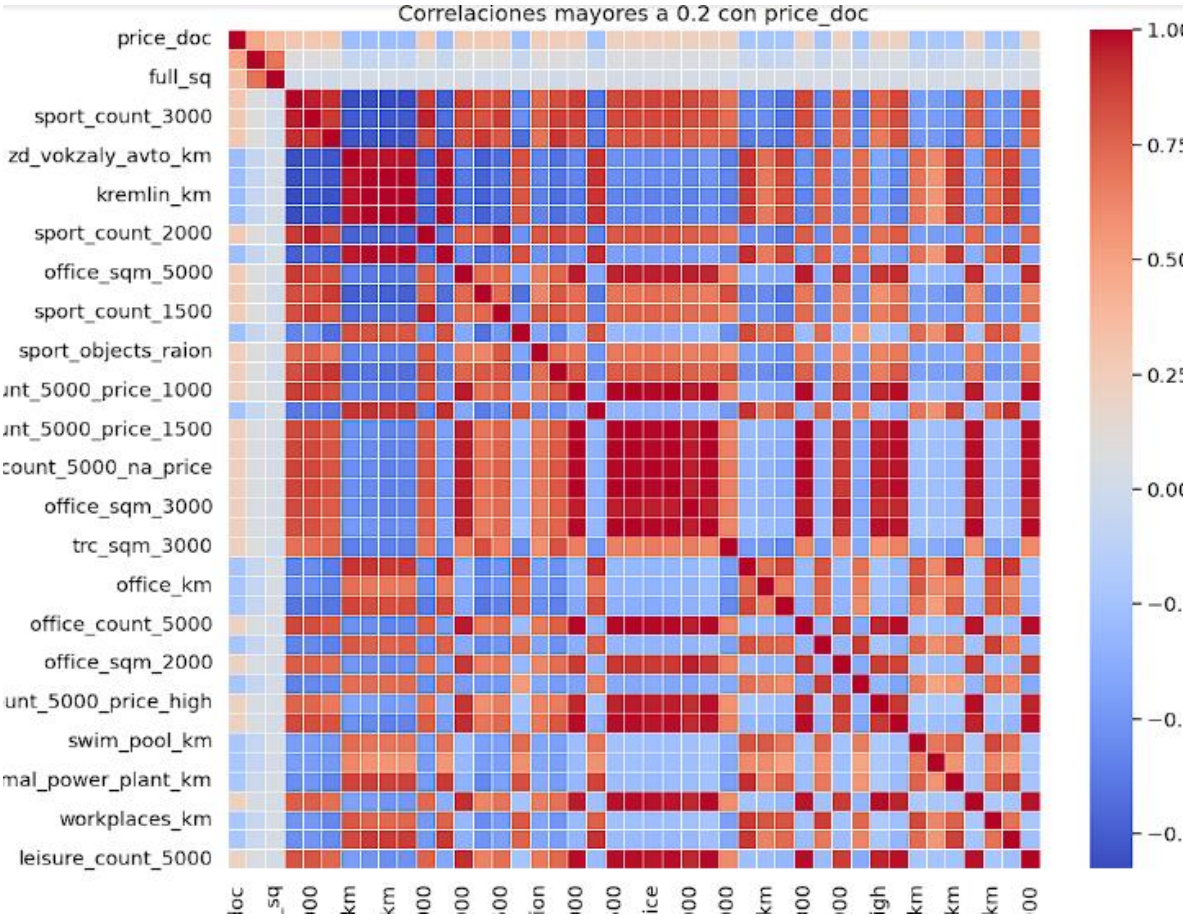
Los tipos de datos incluyen float64 (119 columnas), int64 (157 columnas) y object (16 columnas).

La memoria utilizada por el conjunto de entrenamiento es de aproximadamente 67.9 MB, y para el conjunto de prueba es de aproximadamente 17.0 MB.

Este conjunto de datos proporciona una amplia gama de información que abarca desde detalles específicos de transacciones hasta indicadores macroeconómicos, ofreciendo así una base integral para el análisis y la predicción de los precios de las propiedades en el mercado inmobiliario ruso.

Exploración de datos:

Es muy importante conocer de antemano la correlación de la variables con nuestra variable objetivo:



Descubrimos que realmente eran pocas las variables con correlación significante, de hecho se imprimieron las variables con correlación mayor a 0.2 que suele ser poco, y aun así son pocas las columnas a comparación a la totalidad de columnas

```
Variables con correlación mayor a 0.2:
Index(['price_doc', 'num_room', 'full_sq', 'sport_count_5000',
       'sport_count_3000', 'trc_count_5000', 'zd_vokzaly_avto_km',
       'sadovoe_km', 'kremlin_km', 'bulvar_ring_km', 'sport_count_2000',
       'ttk_km', 'office_sqm_5000', 'trc_sqm_5000', 'sport_count_1500',
       'nuclear_reactor_km', 'sport_objects_raion', 'trc_count_3000',
       'cafe_count_5000_price_1000', 'stadium_km',
       'cafe_count_5000_price_1500', 'cafe_count_5000',
       'cafe_count_5000_na_price', 'cafe_count_5000_price_500',
       'office_sqm_3000', 'cafe_count_5000_price_2500', 'trc_sqm_3000',
       'basketball_km', 'office_km', 'detention_facility_km',
       'office_count_5000', 'university_km', 'office_sqm_2000', 'theater_km',
       'cafe_count_5000_price_high', 'church_count_5000', 'swim_pool_km',
       'catering_km', 'thermal_power_plant_km', 'cafe_count_5000_price_4000',
       'workplaces_km', 'exhibition_km', 'leisure_count_5000'],
      dtype='object')
```

También es importante saber las variables con mayor correlación para luego saber si debemos tomar decisiones significativas con los valores faltantes en el dataset

```
price_doc                  1.000000
num_room                   0.476337
full_sq                    0.341840
sport_count_5000           0.294864
sport_count_3000           0.290651
trc_count_5000             0.289371
zd_vokzaly_avto_km         0.284069
sadovoe_km                 0.283622
kremlin_km                 0.279249
bulvar_ring_km             0.279158
sport_count_2000           0.278056
ttk_km                     0.272620
office_sqm_5000            0.269977
trc_sqm_5000               0.268072
sport_count_1500           0.258376
nuclear_reactor_km         0.257946
sport_objects_raion        0.252794
trc_count_3000             0.242068
cafe_count_5000_price_1000 0.240464
stadium_km                 0.236924
```

Posteriormente, se analizaron las variables numéricas con datos faltantes y se decidió llenarlas con la media de los datos, gracias a que su distribución era relativamente normal y algunas de ellas con la equivalente normal, a continuación veremos las columnas con datos faltantes:

```
life_sq                                  6383
floor                                     167
max_floor                                9572
material                                 9572
build_year                              13605
num_room                                 9572
kitch_sq                                 9572
state                                   13559
preschool_quota                          6688
school_quota                             6685
hospital_beds_raion                     14441
raion_build_count_with_material_info     4991
build_count_block                        4991
build_count_wood                         4991
build_count_frame                        4991
build_count_brick                        4991
build_count_monolith                     4991
build_count_panel                        4991
build_count_foam                         4991
build_count_slag                         4991
build_count_mix                          4991
raion_build_count_with_builddate_info    4991
build_count_before_1920                  4991
build_count_1921-1945                    4991
build_count_1946-1970                    4991
build_count_1971-1995                    4991
build_count_after_1995                   4991
metro_min_walk                             25
metro_km_walk                              25
railroad_station_walk_km                   25
railroad_station_walk_min                  25
ID_railroad_station_walk                   25
cafe_sum_500_min_price_avg               13281
cafe_sum_500_max_price_avg               13281
```

```
cafe_sum_1000_min_price_avg    6524
cafe_sum_1000_max_price_avg    6524
cafe_avg_price_1000            6524
cafe_sum_1500_min_price_avg    4199
cafe_sum_1500_max_price_avg    4199
cafe_avg_price_1500            4199
cafe_sum_2000_min_price_avg    1725
cafe_sum_2000_max_price_avg    1725
cafe_avg_price_2000            1725
cafe_sum_3000_min_price_avg     991
cafe_sum_3000_max_price_avg     991
cafe_avg_price_3000             991
prom_part_5000                  178
cafe_sum_5000_min_price_avg     297
cafe_sum_5000_max_price_avg     297
cafe_avg_price_5000             297
```

Con estos datos faltantes se hizo un análisis para saber cómo iba a funcionar el modelo si eliminábamos las columnas que tienen una gran cantidad de datos faltantes:

```python
def getXY(X):
    # Selecciona solo las columnas numéricas (float e int)
    numeric_columns = X.select_dtypes(include=['float64', 'int64']).columns
    X_numeric = X[numeric_columns]

    y = X["price_doc"].values  # No es necesario convertir a float si los valores son int64
    X_numeric = X_numeric.drop(columns=["price_doc"])  # Elimina la columna objetivo para obtener las variables predictoras

    return X_numeric, y

# Obtén las variables X e y antes de eliminar columnas
X, y = getXY(X)

# Elimina filas con NaN
X = X.dropna()
y = y[X.index]  # Ajusta y para que coincida con las filas remanentes en X

# División de datos antes de eliminar columnas
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Entrenamiento y evaluación del modelo antes de eliminar columnas
model_before_drop = RandomForestRegressor()
model_before_drop.fit(X_train, y_train)
predictions_before_drop = model_before_drop.predict(X_test)
mae_before_drop = mean_absolute_error(y_test, predictions_before_drop)
print(f"MAE Before Drop: {mae_before_drop}")

# Lista de columnas a eliminar
columns_to_drop = [
    'build_year', 'state', 'hospital_beds_raion',
    'cafe_sum_500_min_price_avg', 'cafe_sum_500_max_price_avg', 'cafe_avg_price_500'
]
X_after_drop = X.drop(columns=columns_to_drop)

# Obtén las variables X e y después de eliminar columnas
X_after = X_after_drop
y_after = y[:X_after.shape[0]]  # Ajusta y para que coincida con la longitud de X_after

# División de datos después de eliminar columnas
X_train_after, X_test_after, y_train_after, y_test_after = train_test_split(X_after, y_after, test_size=0.2, random_state=42)

# Entrenamiento y evaluación del modelo después de eliminar columnas
model_after_drop = RandomForestRegressor()
model_after_drop.fit(X_train_after, y_train_after)
predictions_after_drop = model_after_drop.predict(X_test_after)
mae_after_drop = mean_absolute_error(y_test_after, predictions_after_drop)
print(f"MAE After Drop: {mae_after_drop}")
```

```
MAE Before Drop: 2091256.5970057896
MAE After Drop: 2159994.9257154674
```

Dado que el MAE After Drop es más alto que el MAE Before Drop, podemos concluir que la eliminación de esas columnas específicas no benefició al modelo en términos de precisión predictiva. Es posible que la información contenida en esas columnas sea relevante para las predicciones del modelo, y al eliminarlas, se perdió información valiosa. Pese a no tener mucha relación con la variable objetivo, la combinación de estas columnas con otras es necesaria para obtener una mejor predicción

Aunque las características eliminadas pueden no tener una correlación fuerte con la variable objetivo por sí mismas, podrían proporcionar información complementaria junto con otras

características. Su ausencia podría estar afectando la capacidad del modelo para capturar patrones más complejos.

La eliminación de ciertas características podría afectar a las interacciones entre características, especialmente si esas características interactúan con otras para influir en las predicciones del modelo.

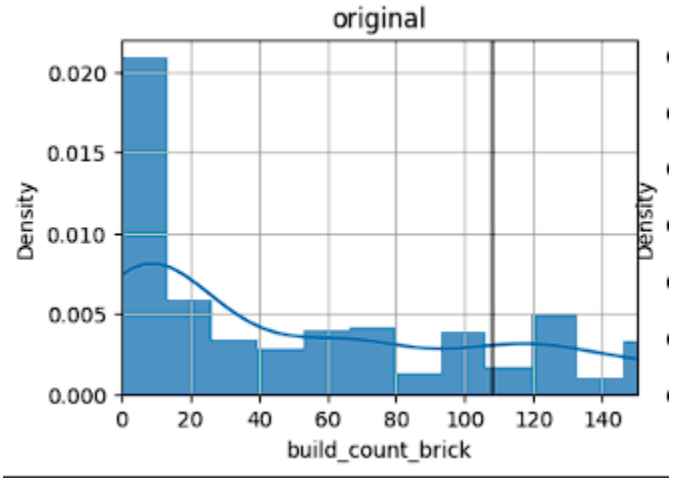
También se implementó un análisis para elegir el método de sustitución de valores faltantes para el total del conjunto de datos:

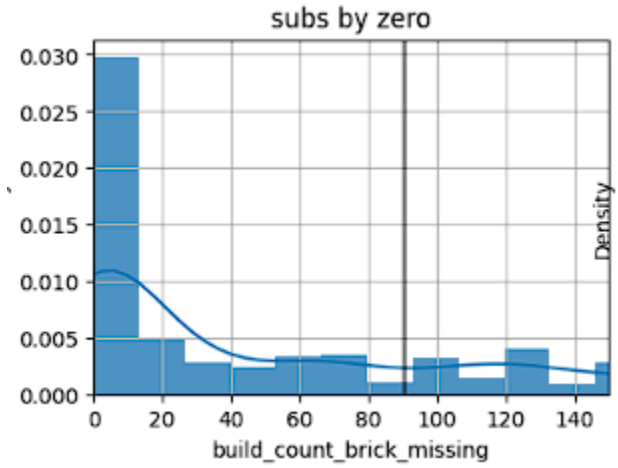
```
from sklearn.model_selection import cross_val_score, ShuffleSplit
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, make_scorer
from scipy.stats import ttest_ind
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

def subs_policies(d, col):
    mcol = "%s_missing"%col
    dn = d.T.dropna().T
    dn = dn[[i for i in dn.columns if d[i].dtype!=object]]
    print (dn.shape)

    na_idxs = np.argwhere(d[col].isna().values)[:,0]

    dl0 = dn.copy()
    dlm = dn.copy()
    dlr = dn.copy()

    dl0[mcol] = d[col].fillna(0)
    dlm[mcol] = d[col].fillna( d[col].mean())

    k = d[col].copy()
    k[k.isna()] = np.random.normal(loc=np.mean(k), scale=np.std(k), size=np.sum(k.isna()))
    dlr[mcol] = k

    f0 = lambda: xdistplot(d[col].dropna(), "original", [0,150])
    f1 = lambda: xdistplot(dl0[mcol], "subs by zero", [0,150])
    f2 = lambda: xdistplot(dlm[mcol], "subs by mean", [0,150])
    f3 = lambda: xdistplot(dlr[mcol], "subs by equivalent normal", [0,150])

    mlutils.figures_grid(4,1, [f0, f1, f2, f3], figsize=(20,3))
    return dn, dl0, dlm, dlr, na_idxs
```

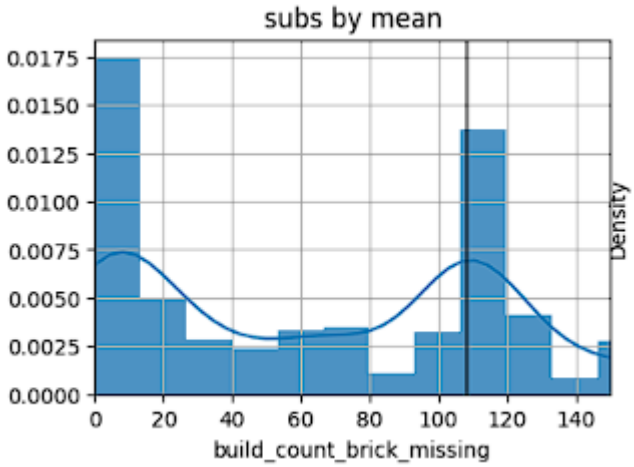
```python
def getXY (dn):
    xcols = [i for i in dn.columns if i!="price_doc"]
    X = dn[xcols].values.astype(float)
    y = dn.price_doc.values.astype(float)
    return X,y,xcols

def experiment(dn, estimator, n_models=8, test_size=.3):
    X,y,_ = getXY(dn)
    r = cross_val_score(estimator, X, y, cv=ShuffleSplit(n_models, test_size=test_size),
                        scoring=make_scorer(mean_absolute_error))
    return r

def HTest(ref_dataset, h_datasets, n_models=10, experiment=experiment, **kwargs):
    estimator = RandomForestRegressor(n_estimators=10)
    re = [experiment(i, estimator, n_models=n_models, **kwargs) for i in pbar([ref_dataset]+h_datasets)]

    for r in re[1:]:
        print (ttest_ind(re[0],r))


dn, dl0, dlm, dlr, na_idxs = subs_policies(X, "build_count_brick")
HTest(dn, [dl0, dlm, dlr], n_models=10)
```

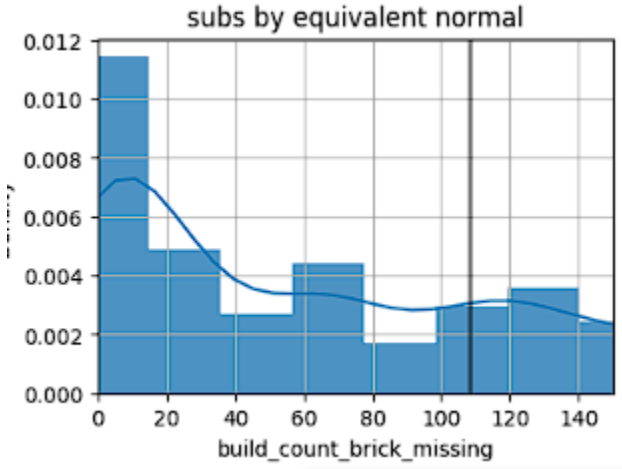
```
TtestResult(statistic=1.5702801306387524, pvalue=0.13376183145028211, df=18.0)
TtestResult(statistic=0.8975450700499882, pvalue=0.3812767064477932, df=18.0)
TtestResult(statistic=1.2385425594079629, pvalue=0.23142464503421625, df=18.0)
```

Se entrenan y evalúan los modelos en todo el conjunto de datos, tanto en registros con datos faltantes como en registros sin datos faltantes, y tambien se comparan las diferentes técnicas de imputación con respecto al rendimiento general del modelo en el conjunto completo.









Los resultados de las pruebas de hipótesis muestran los estadísticos t y los valores p para comparar el rendimiento de los modelos entre el grupo de control y el grupo de población. En este caso, los p-values no son significativos, lo que sugiere que, en este escenario simplificado, ninguna de las políticas de sustitución proporciona una mejora general en el rendimiento del modelo. Además, se señala que los experimentos repetidos no muestran

evidencia de que alguna de las aproximaciones sea mejor que las demás en términos de p-value. En resumen, no hay suficiente evidencia para concluir que una técnica de sustitución es mejor que otra en este contexto particular.

Análisis para el llenado de valores faltantes solo en registros con valores faltantes:

Se entrenan y evalúan los modelos solo en el subconjunto de datos que contiene registros con datos faltantes y también se comparan las diferentes técnicas de imputación solo en los registros con datos faltantes, ignorando los registros completos.

```python
def na_cross_val_score(estimator, X, y, cv, scoring, val_idxs):
    r = []
    for tr_idxs, ts_idxs in cv.split(X):
        tr_idxs, ts_idxs = np.r_[tr_idxs], np.r_[ts_idxs]
        rf.fit(X[tr_idxs], y[tr_idxs])
        valts_idxs = np.r_[[i for i in ts_idxs if i in val_idxs]]
        r.append(scoring(rf, X[valts_idxs], y[valts_idxs]))
    return r


def na_experiment(dn, estimator, na_idxs, n_models=5, test_size=.3):
    X,y,_ = getXY(dn)
    r = na_cross_val_score(estimator, X, y, cv=ShuffleSplit(n_models, test_size=test_size),
                           scoring=make_scorer(mean_absolute_error), val_idxs=na_idxs)
    return r

## KEEPOUTPUT
HTest(dn, [dl0, dlm, dlr], experiment=na_experiment, na_idxs=na_idxs, n_models=8)
```

```
TtestResult(statistic=-0.2901042602418596, pvalue=0.7759890428698888, df=14.0)
TtestResult(statistic=0.6983287711623363, pvalue=0.49640827865167014, df=14.0)
TtestResult(statistic=-2.1001107909927677, pvalue=0.054322010429272, df=14.0)
```

Técnica de sustitución por cero (dl0):

El estadístico t es -0.29 con un p-value de 0.776.

Interpretación: No hay suficiente evidencia para rechazar la hipótesis nula. En este enfoque específico de evaluación (considerando solo registros con datos faltantes), la sustitución por cero no parece tener un impacto significativo en el rendimiento del modelo.

Técnica de sustitución por la media (dlm):

El estadístico t es 0.70 con un p-value de 0.496.

Interpretación: No hay suficiente evidencia para rechazar la hipótesis nula. Al igual que con la sustitución por cero, la sustitución por la media no muestra un impacto significativo en el rendimiento del modelo en este contexto específico.

Técnica de sustitución por distribución normal equivalente (dlr):

El estadístico t es -2.10 con un p-value de 0.054.

Interpretación: Aunque el p-value está cerca de 0.05, aún no es lo suficientemente bajo como para rechazar la hipótesis nula de manera concluyente. Sin embargo, hay una tendencia hacia el rechazo de la hipótesis nula, lo que sugiere que la sustitución por una distribución normal equivalente podría tener un impacto positivo en el rendimiento del modelo en este escenario específico.

En las variables categóricas se realizó una codificación one-hot para las siguientes columnas y valores:

```
product_type ['Investment' 'OwnerOccupier']
  sub_area ['Ajeroport' 'Akademicheskoe' 'Alekseevskoe' "Altuf'evskoe" 'Arbat
 'Babushkinskoe' 'Basmannoe' 'Begovoe' 'Beskudnikovskoe' 'Bibirevo'
 'Birjulevo Vostochnoe' 'Birjulevo Zapadnoe' 'Bogorodskoe' 'Brateevo'
 'Butyrskoe' 'Caricyno' 'Cheremushki' "Chertanovo Central'noe"
 'Chertanovo Juzhnoe' 'Chertanovo Severnoe' 'Danilovskoe' 'Dmitrovskoe'
 'Donskoe' 'Dorogomilovo' 'Filevskij Park' 'Fili Davydkovo' 'Gagarinskoe'
 "Gol'janovo" 'Golovinskoe' 'Hamovniki' 'Horoshevo-Mnevniki'
 'Horoshevskoe' 'Hovrino' 'Ivanovskoe' 'Izmajlovo' 'Jakimanka'
 'Jaroslavskoe' 'Jasenevo' 'Juzhnoe Butovo' 'Juzhnoe Medvedkovo'
 'Juzhnoe Tushino' 'Juzhnoportovoe' 'Kapotnja' "Kon'kovo" 'Koptevo'
 'Kosino-Uhtomskoe' 'Kotlovka' "Krasnosel'skoe" 'Krjukovo' 'Krylatskoe'
 'Kuncevo' 'Kurkino' "Kuz'minki" 'Lefortovo' 'Levoberezhnoe' 'Lianozovo'
 'Ljublino' 'Lomonosovskoe' 'Losinoostrovskoe' "Mar'ina Roshha" "Mar'ino"
 'Marfino' 'Matushkino' 'Meshhanskoe' 'Metrogorodok' 'Mitino'
 'Molzhaninovskoe' "Moskvorech'e-Saburovo" 'Mozhajskoe'
 'Nagatino-Sadovniki' 'Nagatinskij Zaton' 'Nagornoe' 'Nekrasovka'
 'Nizhegorodskoe' 'Novo-Peredelkino' 'Novogireevo' 'Novokosino'
 'Obruchevskoe' 'Ochakovo-Matveevskoe' 'Orehovo-Borisovo Juzhnoe'
 'Orehovo-Borisovo Severnoe' 'Ostankinskoe' 'Otradnoe' 'Pechatniki'
 'Perovo' 'Pokrovskoe Streshnevo' 'Poselenie Desjonovskoe'
 'Poselenie Filimonkovskoe' 'Poselenie Kievskij' 'Poselenie Klenovskoe'
 'Poselenie Kokoshkino' 'Poselenie Krasnopahorskoe'
 'Poselenie Marushkinskoe' 'Poselenie Mihajlovo-Jarcevskoe'
 'Poselenie Moskovskij' 'Poselenie Mosrentgen' 'Poselenie Novofedorovskoe'
 'Poselenie Pervomajskoe' 'Poselenie Rjazanovskoe' 'Poselenie Rogovskoe'
 'Poselenie Shhapovskoe' 'Poselenie Shherbinka' 'Poselenie Sosenskoe'
 'Poselenie Vnukovskoe' 'Poselenie Voronovskoe' 'Poselenie Voskresenskoe'
 'Preobrazhenskoe' 'Presnenskoe' 'Prospekt Vernadskogo' 'Ramenki'
 'Rjazanskij' 'Rostokino' 'Savelki' 'Savelovskoe' 'Severnoe'
 'Severnoe Butovo' 'Severnoe Izmajlovo' 'Severnoe Medvedkovo'
 'Severnoe Tushino' 'Shhukino' 'Silino' 'Sokol' "Sokol'niki"
 'Sokolinaja Gora' 'Solncevo' 'Staroe Krjukovo' 'Strogino' 'Sviblovo'
 'Taganskoe' "Tekstil'shhiki" 'Teplyj Stan' 'Timirjazevskoe'
 'Troickij okrug' 'Troparevo-Nikulino' 'Tverskoe' 'Veshnjaki' 'Vnukovo'
 'Vojkovskoe' 'Vostochnoe' 'Vostochnoe Degunino' 'Vostochnoe Izmajlovo'
 'Vyhino-Zhulebino' "Zamoskvorech'e" 'Zapadnoe Degunino' 'Zjablikovo'
 'Zjuzino']
culture_objects_top_25 ['no' 'yes']
```

```
culture_objects_top_25 ['no' 'yes']
thermal_power_plant_raion ['no' 'yes']
incineration_raion ['no' 'yes']
oil_chemistry_raion ['no' 'yes']
radiation_raion ['no' 'yes']
railroad_terminal_raion ['no' 'yes']
big_market_raion ['no' 'yes']
nuclear_reactor_raion ['no' 'yes']
detention_facility_raion ['no' 'yes']
water_1line ['no' 'yes']
big_road1_1line ['no' 'yes']
railroad_1line ['no' 'yes']
   ecology ['excellent' 'good' 'no data' 'poor' 'satisfactory']
```

Finalmente aplicamos todos los cambios en un método que se pudiera usar para los datos de test y como dato adicional, tuve problemas con respecto a tiempo de procesamiento por la gran cantidad de dimensiones y datos. También el conjunto de train había generado una columna más, ya que la variable categórica sub_area tenía una categoría que no estaba contenida en train.

Desarrollo:

En la competición de kaggle decía la implementación de la métrica de desempeño RMSLE (Root Mean Squared Logarithmic Error) es una métrica de evaluación comúnmente utilizada en problemas de regresión, como lo es mi caso. La RMSLE mide la discrepancia entre los valores reales y los valores predichos, con un enfoque particular en la magnitud de los errores relativos.

La fórmula de RMSLE se expresa como:

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

La RMSLE penaliza de manera más suave los errores en predicciones de baja magnitud, lo cual puede ser útil en situaciones donde la variación en los errores no debería ser tratada de manera lineal.

Algunos puntos clave son:

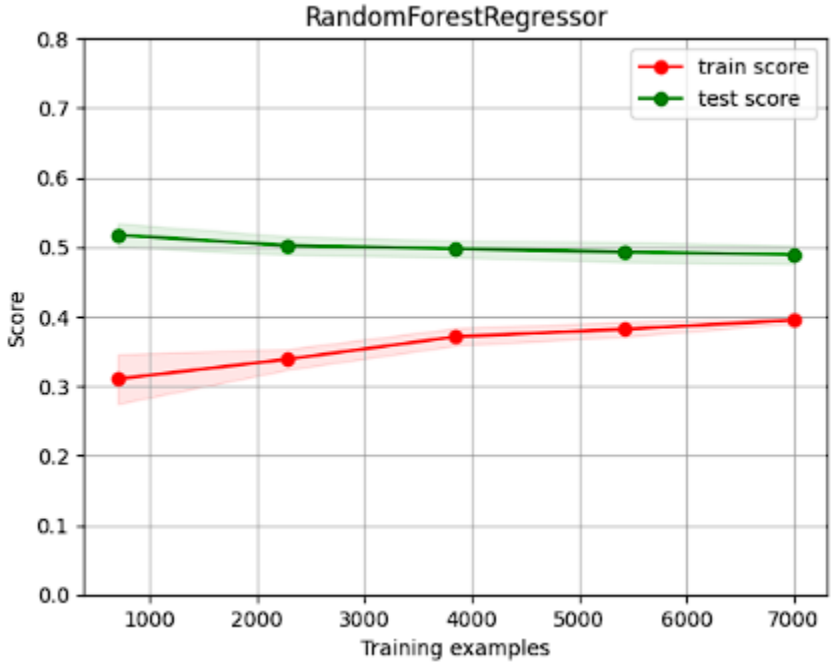
Escalado Logarítmico: utiliza el logaritmo de las predicciones y los valores reales, lo que significa que la escala logarítmica atenúa la influencia de los errores grandes.

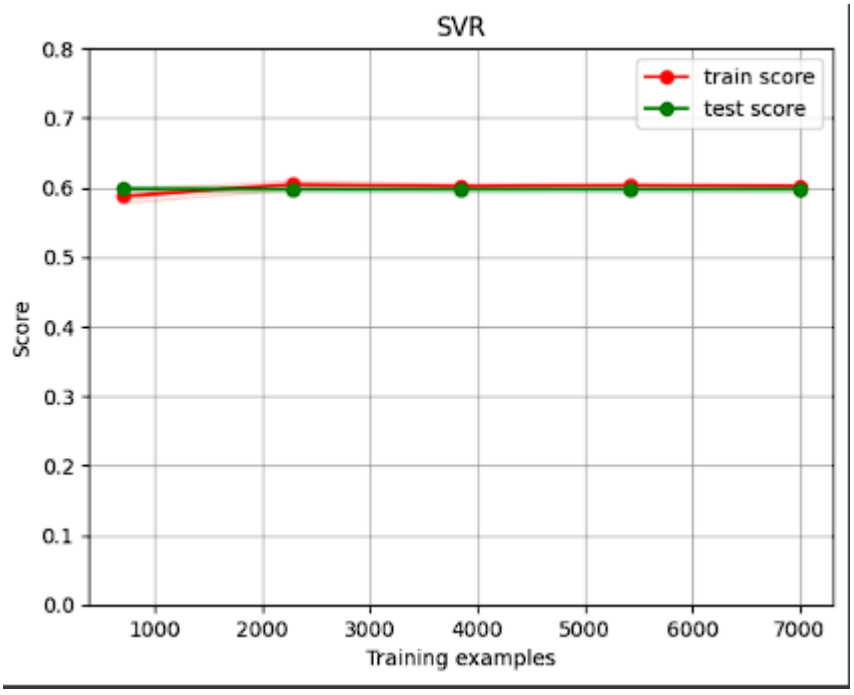
Raíz Cuadrada: La raíz cuadrada se toma al final para que la métrica esté en la misma escala que la variable objetivo (en este caso, el precio de la propiedad).

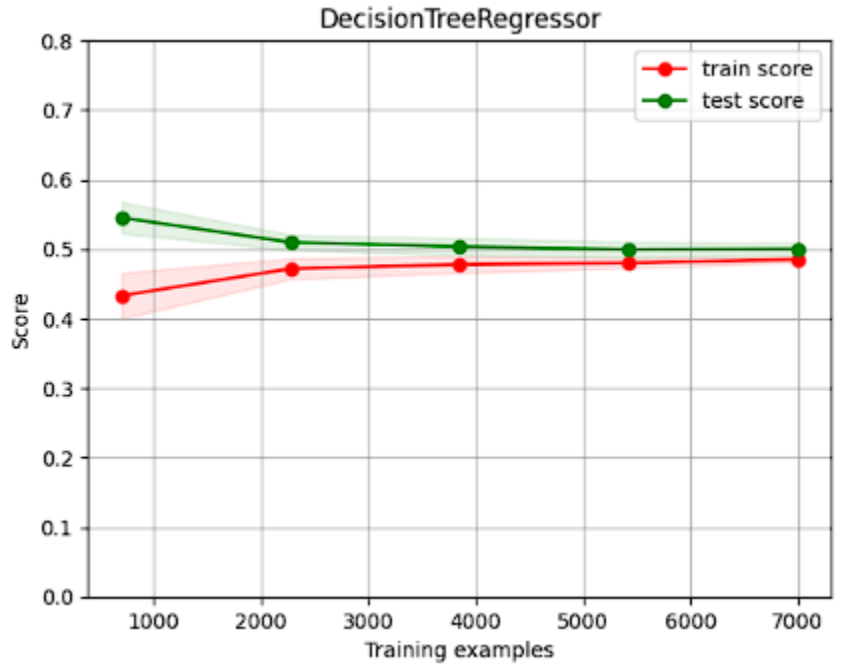
Añadir 1: Se suma 1 en el logaritmo para evitar problemas cuando se toma el logaritmo de cero.

Menor Sensibilidad a Outliers: Al tener en cuenta el logaritmo, la métrica es menos sensible a los outliers, lo que significa que los errores en las predicciones de precios extremadamente altos o bajos no afectan demasiado la métrica.

Ahora bien, para los modelos supervisados se hizo el análisis con 3 hiperparametros, para la implementación se escogieron 1000 datos aleatorios de entrenamiento para hacer las posteriores divisiones de train y test (se hizo de ese modo, por las previas iteraciones con tiempos de ejecución extremadamente altos):







En conclusión, tras evaluar el rendimiento de tres algoritmos supervisados para predecir el precio de propiedades utilizando la métrica RMSLE, se observa que cada modelo presenta diferentes fortalezas y debilidades en términos de precisión predictiva.

El Random Forest ha destacado como el modelo más robusto, exhibiendo una puntuación RMSLE promedio de aproximadamente 0.482 con una desviación estándar de alrededor de

0.0151. Esta consistencia sugiere una capacidad confiable para realizar predicciones precisas y estables en diferentes divisiones de datos de prueba.

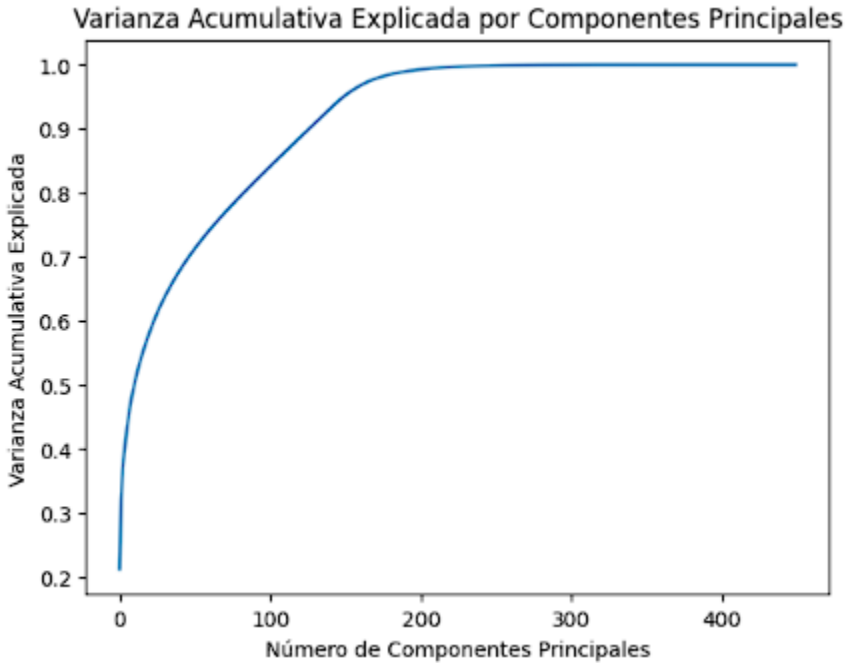
El Support Vector Machine (SVM) ha mostrado un rendimiento decente con una puntuación RMSLE promedio de alrededor de 0.600 y una desviación estándar de aproximadamente 0.0091. Aunque ligeramente menos preciso que el Random Forest, el SVM podría ser una opción viable dependiendo de otros factores, como la interpretabilidad del modelo.

Por último, el Decision Tree ha obtenido un rendimiento intermedio con una puntuación RMSLE promedio de alrededor de 0.500 y una desviación estándar de aproximadamente 0.0145. Aunque proporciona resultados aceptables, podría beneficiarse de técnicas de mejora o consideración de modelos más complejos para optimizar su precisión.

Random Forest parece fue la opción mas aceptable para llevar el modelo a producción, aunque no tuvo un gran desempeño y también se escoge, porque los tiempos de respuesta de los demás parámetros son muy altos.

Para los modelos no supervisados con supervisados se uso PCA con random forest y K-means y la regresesion lineal y random forest. Para decidir el numero de componentes optimo se realizo el siguiente analisis:

```
Número de Componentes: 1, Varianza Acumulativa Explicada: 21.25%
Número de Componentes: 21, Varianza Acumulativa Explicada: 58.30%
Número de Componentes: 41, Varianza Acumulativa Explicada: 67.91%
Número de Componentes: 61, Varianza Acumulativa Explicada: 74.27%
Número de Componentes: 81, Varianza Acumulativa Explicada: 79.42%
Número de Componentes: 101, Varianza Acumulativa Explicada: 84.13%
Número de Componentes: 121, Varianza Acumulativa Explicada: 88.68%
Número de Componentes: 141, Varianza Acumulativa Explicada: 93.18%
Número de Componentes: 161, Varianza Acumulativa Explicada: 96.77%
Número de Componentes: 181, Varianza Acumulativa Explicada: 98.50%
Número de Componentes: 201, Varianza Acumulativa Explicada: 99.26%
Número de Componentes: 221, Varianza Acumulativa Explicada: 99.64%
Número de Componentes: 241, Varianza Acumulativa Explicada: 99.82%
Número de Componentes: 261, Varianza Acumulativa Explicada: 99.91%
Número de Componentes: 281, Varianza Acumulativa Explicada: 99.96%
Número de Componentes: 301, Varianza Acumulativa Explicada: 99.98%
Número de Componentes: 321, Varianza Acumulativa Explicada: 100.00%
Número de Componentes: 341, Varianza Acumulativa Explicada: 100.00%
Número de Componentes: 361, Varianza Acumulativa Explicada: 100.00%
Número de Componentes: 381, Varianza Acumulativa Explicada: 100.00%
Número de Componentes: 401, Varianza Acumulativa Explicada: 100.00%
Número de Componentes: 421, Varianza Acumulativa Explicada: 100.00%
Número de Componentes: 441, Varianza Acumulativa Explicada: 100.00%
```



Aun escogiendo una dimensionalidad menor, los resultados no fueron muy óptimos, los modelos no supervisados no ayudaron a mejorar el rendimiento de los modelos.

Retos y consideraciones de despliegue:

Durante el proceso de desarrollo y evaluación de modelos, se enfrentaron diversos retos y se tomaron consideraciones clave que influyen en el despliegue exitoso de la solución. Estos desafíos ofrecen una visión valiosa sobre las complejidades inherentes al proyecto y permiten aprender de las iteraciones menos exitosas.

Uno de los desafíos más significativos fue la gestión de datos faltantes en el conjunto de entrenamiento. A pesar de los esfuerzos para imputar valores ausentes, algunas variables críticas presentaron una cantidad considerable de datos faltantes, lo que afectó directamente la capacidad del modelo para realizar predicciones precisas. Este desafío resalta la importancia de abordar de manera efectiva la imputación de datos en etapas tempranas del proyecto.

Se observó que ciertos modelos, especialmente el Support Vector Machine, son sensibles a la configuración de parámetros. Aunque se realizaron ajustes sistemáticos, la búsqueda de la combinación óptima de parámetros sigue siendo un desafío continuo. Esto subraya la necesidad de explorar exhaustivamente el espacio de hiperparámetros y considerar estrategias avanzadas, para mejorar la robustez del modelo.

La complejidad inherente de algunos modelos, como el Random Forest, puede dificultar la interpretación de las decisiones del modelo. Este aspecto puede presentar un desafío en entornos donde la transparencia y la comprensión de los resultados son críticas. Explorar

enfoques para explicar las predicciones de modelos complejos, se identificó como una consideración importante.

A la hora de poner los modelos predictivos en producción es fundamental contar con un plan de monitoreo para poder supervisar el comportamiento, resultados y estado del modelo. De esta forma es posible detectar problemas con el modelo y poder realizar ajustes o actualizaciones en este ya que es necesario tener una retroalimentación por parte de los modelos ejecutados.

Conclusiones:

En conclusión, el proceso de desarrollo de modelos para predecir el precio de propiedades ha sido un viaje caracterizado por la resolución de retos y la adaptación continua. Aunque no todas las iteraciones lograron resultados sobresalientes, cada una contribuyó al aprendizaje y proporcionó información valiosa.

Referencias:

https://github.com/JohanSH7/PROYECTO_SBERBANK_RUSSIAN_HOUSING_MARKET

https://www.kaggle.com/competitions/sberbank-russian-housing-market/data