

BIG DATA

PROYECTO FINAL

Jorge González Piedra



Jorge González Piedra

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

Fecha: 05, FEB, 2023



DESARROLLO

1. Un informe científico, en el que se transmitan los resultados de los análisis realizados. Aquí explicaremos paso a paso cada uno de los apartados con las conclusiones correspondientes de las tareas realizadas. Podremos incluir secciones de código si es necesario y por supuesto, los resultados de cada una de las tareas realizadas sobre los datos obtenidos a través de la ejecución del código contenido en el documento técnico.
2. Un documento técnico que tendrá el código fuente (PySpark) empleado para la resolución de cada una de las tareas. El código fuente debe ser insertado como imágenes y con un tamaño que permita leer el texto contenido en las imágenes.
3. Una presentación guardada en formato pdf. Esta presentación nos servirá para mostrar los resultados de cada una de las tareas y no contendrá código fuente sino que mostrará los resultados obtenidos siguiendo las guías de presentación que hemos visto en el módulo de proyectos *big data* y *storytelling*.



1. Cassandra

1.1. Recuperar registros de la aerolínea "Air China"

```
token@cqlsh:airtrafic> select * from airtrafic.airtraffic_table WHERE Op-  
erating_Airline = 'Air China' ALLOW FILTERING;
```

Los datos se encuentran adjuntos a la entrega en la carpeta
airchina_data_extraction

1.2. Recuperar todos los vuelos de la compañía "Air Berlín" embarcados por la puerta "G"

```
token@cqlsh:airtrafic> select * from airtrafic.airtraffic_table WHERE Op-  
erating_Airline = 'Air Berlin' AND boarding_area = 'G' ALLOW FILTERING;
```

Los datos se encuentran adjuntos a la entrega en la carpeta
airberlin_data_extraction

2. PySpark

2.1. Creación del Dataframe y análisis preliminar

2.1.1. Creación dataframe

```
from pyspark.sql.functions import col

df_airport = spark.read.options(inferSchema='True', delimiter=',',
header=True).csv("/content/drive/MyDrive/TOKIO/Big Data - Cloud Computing/01 - Big Data/PROYECTO FINAL/Air_Traffic_Passenger_Statistics.csv")
```

```
df_airport.printSchema()
```

```
root
|-- Activity Period: integer (nullable = true)
|-- Operating Airline: string (nullable = true)
|-- Operating Airline IATA Code: string (nullable = true)
|-- Published Airline: string (nullable = true)
|-- Published Airline IATA Code: string (nullable = true)
|-- GEO Summary: string (nullable = true)
|-- GEO Region: string (nullable = true)
|-- Activity Type Code: string (nullable = true)
|-- Price Category Code: string (nullable = true)
|-- Terminal: string (nullable = true)
|-- Boarding Area: string (nullable = true)
|-- Passenger Count: integer (nullable = true)
|-- Adjusted Activity Type Code: string (nullable = true)
|-- Adjusted Passenger Count: integer (nullable = true)
|-- Year: integer (nullable = true)
|-- Month: string (nullable = true)
```

2.1.2. Seleccionar compañías diferentes en el fichero

```
df_airport.dropDuplicates(["OperatingAirline"]).select("OperatingAir-  
line").count()  
df_airport.dropDuplicates(["OperatingAirline"]).select("OperatingAir-  
line").show()
```

2.1.3. Pasajeros de media de los vuelos de cada compañía

```
df_airport.groupBy("OperatingAirline").mean("PassengerCount", "Adjusted-  
PassengerCount").show()
```

2.1.4. Eliminación registros duplicados por GEO Region

```
df_GEORegion_no_duplicates = spark.sql("select a1.* FROM " \  
"df_airport_view a1, " \  
(SELECT GEORegion, MAX(PassengerCount) PassengerCount FROM df_air-  
port_view GROUP BY GEORegion) a2 "\  
"WHERE a1.GEORegion = a2.GEORegion " \  
"AND a1.PassengerCount = a2.PassengerCount")  
df_GEORegion_no_duplicates.show();
```

2.1.5. Volcar resultados

```
df_GEORegion_no_duplicates.write.options(header="True").csv("/con-  
tent/drive/MyDrive/TOKIO/Big Data - Cloud Computing/01 - Big Data/PROY-  
ECTO FINAL/Entrega/Ficheros/airtraffic_drop_duplicates_georegion")
```

Los datos se encuentran adjuntos a la entrega en el fichero *airtraffic_drop_duplicates_georegion.csv*.

2.2. Análisis estadístico

2.2.1. Análisis descriptivo

2.2.1.1. Activity Period

```
[21] df_airport.describe("ActivityPeriod").show()
```

summary	ActivityPeriod
count	15007
mean	201045.07336576266
stddev	313.33619609986414
min	200507
max	201603

```
[41] df_airport_pandas['ActivityPeriod'].mode()
```

```
0    200807  
dtype: int32
```

2.2.1.2. Operating airline

```
df_airport_pandas['OperatingAirline'].mode()
```

```
0    United Airlines - Pre 07/01/2013  
dtype: object
```

2.2.1.3. Operating airline IATA code

```
df_airport_pandas['OperatingAirlineIATACode'].mode()
```

```
0    UA  
dtype: object
```

2.2.1.4. Published airline

```
df_airport_pandas['PublishedAirline'].mode()  
  
0    United Airlines - Pre 07/01/2013  
dtype: object
```

2.2.1.5. Published airline IATA code

```
df_airport_pandas['PublishedAirlineIATACode'].mode()  
  
0    UA  
dtype: object
```

2.2.1.6. GEO summary

```
df_airport_pandas['GEOSummary'].mode()  
  
0    International  
dtype: object
```

2.2.1.7. GEO Region

```
df_airport_pandas['GEORegion'].mode()  
  
0    US  
dtype: object
```

2.2.1.8. Activity type code


```
df_airport.select("ActivityTypeCode").dropDuplicates().show()
```

```
+-----+
|ActivityTypeCode|
+-----+
|      Enplaned|
|  Thru / Transit|
|      Deplaned|
+-----+
```

2.2.1.9. Price category code

```
df_airport_pandas['PriceCategoryCode'].mode()
```

```
0    Other
dtype: object
```

2.2.1.10. Terminal

```
df_airport_pandas['Terminal'].mode()
```

```
0    International
dtype: object
```

2.2.1.11. Boarding area

```
df_airport_pandas['BoardingArea'].mode()
```

```
0    A
dtype: object
```

2.2.1.12. Passenger count

```
df_airport.describe("PassengerCount").show()
```

```
+-----+-----+
|summary| PassengerCount|
+-----+-----+
|  count|           15007|
|   mean| 29240.521090157927|
|  stddev| 58319.509284123524|
|    min|                1|
|    max|           659837|
+-----+-----+
```

2.2.1.13. Adjusted activity type code

```
df_airport_pandas['AdjustedActivityTypeCode'].mode()
```

```
0    Deplaned
dtype: object
```

2.2.1.14. Adjusted Passenger count

```
df_airport.describe("AdjustedPassengerCount").show()
```

```
+-----+-----+
|summary| AdjustedPassengerCount|
+-----+-----+
|  count|           15007|
|   mean| 29331.917105350836|
|  stddev|  58284.1822186625|
|    min|                1|
|    max|           659837|
+-----+-----+
```

2.2.1.15. Year

```
[62] df_airport.describe("Year").show()
```

summary	Year
count	15007
mean	2010.385220230559
stddev	3.137589043169972
min	2005
max	2016

```
[63] df_airport_pandas['Year'].mode()
```

```
0    2015  
dtype: int32
```

2.2.1.16. Month

```
df_airport_pandas['Month'].mode()
```

```
0    August  
dtype: object
```

2.2.2. Análisis de correlación

```
df_airport_pd.corr()
```

```
df_airport_pd.corr().style.background_gradient(cmap='coolwarm')
```

	ActivityPeriod	OperatingAirline	OperatingAirlineIATAcode	PublishedAirline	PublishedAirlineIATAcode	GEOSummary	GEORegion	ActivityTypeCode	PriceCategoryCode	Terminal	BoardingArea	PassengerCount	AdjustedActivityTypeCode	AdjustedPassengerCount	Year	Month
ActivityPeriod	1.000000	0.008132	-0.043771	0.009796	-0.018936	0.066247	-0.028159	-0.052887	-0.006257	-0.008901	-0.005209	0.061871	-0.052887	0.060889	0.995763	-0.002989
OperatingAirline	0.008132	1.000000	0.823688	0.968828	0.818090	-0.130956	0.151120	0.106644	-0.006112	0.197959	0.251975	0.185424	0.106644	0.108427	0.006183	-0.00452
OperatingAirlineIATAcode	-0.043771	0.823688	1.000000	0.790621	0.919021	-0.131919	0.101535	0.099556	-0.091939	0.208314	0.280294	0.122244	0.099556	0.123170	-0.043257	0.000661
PublishedAirline	0.009796	0.968828	0.790621	1.000000	0.859995	-0.083020	0.108379	0.098414	-0.095284	0.199894	0.275990	0.200862	0.098414	0.201890	0.009641	0.000609
PublishedAirlineIATAcode	-0.010936	0.818090	0.919021	0.859995	1.000000	-0.027591	0.008866	0.097818	-0.105385	0.167910	0.312836	0.155368	0.097818	0.156337	-0.010837	0.001689
GEOSummary	0.066247	-0.130956	-0.131919	-0.083020	-0.027591	1.000000	-0.871826	-0.026760	0.411498	0.574422	0.109553	-0.395743	-0.026760	-0.396856	0.068045	-0.001139
GEORegion	-0.028159	0.151120	0.101535	0.108379	0.008866	-0.871826	1.000000	0.033899	-0.382864	0.509119	-0.121033	0.336113	0.033899	0.336880	-0.028129	0.000949
ActivityTypeCode	-0.052887	0.106644	0.099556	0.098414	0.097818	-0.026760	0.033899	1.000000	0.001004	0.087788	0.087706	-0.071423	1.000000	-0.067804	-0.052364	-0.001523
PriceCategoryCode	-0.006257	-0.006112	-0.091939	-0.095284	-0.105385	0.411498	-0.382864	0.001004	1.000000	-0.102936	0.213485	-0.055047	0.001004	-0.064661	-0.095683	-0.003627
Terminal	-0.008901	0.197959	0.208314	0.199894	0.167910	-0.574422	0.509119	0.087788	-0.102936	1.000000	0.168414	0.429146	0.087788	0.430687	-0.008155	-0.000693
BoardingArea	-0.005209	0.251975	0.280294	0.275990	0.312936	0.109553	-0.121033	0.087706	0.213485	0.168414	1.000000	0.131091	0.087706	0.132147	-0.005109	-0.000581
PassengerCount	0.061871	0.185424	0.122244	0.200862	0.155368	-0.395743	0.336113	-0.071423	-0.065847	0.429146	0.131091	1.000000	-0.071423	0.999941	0.060069	0.000413
AdjustedActivityTypeCode	-0.052887	0.106644	0.099556	0.098414	0.097818	-0.026760	0.033899	1.000000	0.001004	0.087788	0.087706	-0.071423	1.000000	-0.067804	-0.052364	-0.001523
AdjustedPassengerCount	0.060889	0.108427	0.123170	0.201890	0.156337	-0.396856	0.336980	-0.067804	-0.064661	0.430687	0.132147	0.999941	-0.067804	1.000000	0.050096	0.000365
Year	0.995763	0.006183	-0.043257	0.009641	-0.010837	0.066046	-0.028129	-0.052364	-0.005683	-0.008155	-0.005109	0.000609	-0.052364	0.059096	1.000000	-0.030413
Month	-0.002989	-0.00452	0.000661	0.000609	0.001689	-0.001139	0.000949	-0.001523	-0.003627	-0.000693	-0.000581	0.000413	-0.001523	0.000365	-0.030413	1.000000

Correlaciones mas fuertes:

2.2.2.1. GEO Summary – Price Category Code

```
[ ] df_airport_pd.GEOSummary.corr(df_airport_pd.PriceCategoryCode)
```

```
0.41149848056451377
```

2.2.2.2. GEO Summary – GEO Region

```
[ ] df_airport_pd.GEOSummary.corr(df_airport_pd.GEORegion)
```

```
-0.8718261857198394
```

2.2.2.3. GEO Region – Terminal

```
[ ] df_airport_pd.GEORegion.corr(df_airport_pd.Terminal)
```

```
0.5091186306605863
```

2.2.2.4. GEO Region – Price Category Code

```
[ ] df_airport_pd.PriceCategoryCode.corr(df_airport_pd.GEORegion)

-0.3828639102138204
```

2.2.2.5. Adjusted Passenger Count – GEO Region

```
[ ] df_airport_pd.AdjustedPassengerCount.corr(df_airport_pd.GEORegion)

0.3369804846146507
```

2.2.2.6. Adjusted Passenger Count – Terminal

```
[ ] df_airport_pd.AdjustedPassengerCount.corr(df_airport_pd.Terminal)

0.43068707562529646
```

Utilizamos el **método Point-Biserial** para calcular de manera más precisa la correlación entre las variables dicotómicas y el número de pasajeros.

2.2.2.7. GEO Summary– Adjusted passenger count

```
[58] stats.pointbiserialr(df_airport_pd['GEOSummary'], df_airport_pd['AdjustedPassengerCount'])

PointbiserialrResult(correlation=-0.39685620097984975, pvalue=0.0)
```

2.2.2.8. Price category code – Adjusted passenger count

```
[59] stats.pointbiserialr(df_airport_pd['PriceCategoryCode'], df_airport_pd['AdjustedPassengerCount'])

PointbiserialrResult(correlation=-0.0646612429860395, pvalue=2.2120528625642906e-15)
```

2.2.3. Regresión lineal

- Creación del nuevo DataFrame para estudiar el número de pasajeros a lo largo de los años

```
from pyspark.sql.functions import sum
df_pass_by_year = df_airport.groupBy("Year" , "Month").agg(sum("Passen-
gerCount").alias("PassengerCountSum"))
df_pass_by_year.show()
```

```
+----+-----+-----+
|Year|  Month|PassengerCountSum|
+----+-----+-----+
|2006|September|      2720100|
|2007|    May|      3056934|
|2012|February|      2998119|
|2008|  April|      3029021|
|2006|October|      2834959|
|2011|November|      3326859|
|2006|February|      2223024|
|2014|    July|      4499221|
|2011|  August|      3917884|
|2007|December|      2903637|
|2013|  August|      4347059|
|2009|February|      2359800|
|2014|    May|      4147096|
|2011|October|      3602455|
|2006|    July|      3227605|
|2006|November|      2653887|
|2014|November|      3628786|
|2009|    May|      3177100|
|2013|December|      3814984|
|2014|December|      3855835|
+----+-----+-----+
only showing top 20 rows
```

- Conversión de los valores de la columna *Month* a tipo numérico

```
df_pass_by_year_pd = df_pass_by_year.toPandas()
```

```
import calendar as cal

lower_ma = [m.lower() for m in cal.month_name]
df_pass_by_year_pd['Month'] =
df_pass_by_year_pd['Month'].str.lower().map(lambda m: lower_ma.index(m)).astype('Int8')
```

```
df_pass_by_year_pd.sort_values(by=['Year', 'Month'])
```



	Year	Month	PassengerCountSum
60	2005	7	3225769
62	2005	8	3195866
90	2005	9	2740553
117	2005	10	2770715
84	2005	11	2617333
...
109	2015	11	4013814
82	2015	12	4129052
113	2016	1	3748529
85	2016	2	3543639
31	2016	3	4137679

129 rows × 3 columns

- Creación de una nueva columna para unir Year y Month en un campo de tipo fecha:

```
df_pass_by_year_pd['Date'] = df_pass_by_year_pd[df_pass_by_year_pd.columns[0:2]].apply(lambda x: "-".join(x.values.astype(str)),axis="columns")
df_pass_by_year_pd['Date']=
pd.to_datetime(df_pass_by_year_pd['Date']).dt.strftime("%Y-%m")
df_pass_by_year_pd.sort_values(by=["Date"])
```

	Year	Month	PassengerCountSum	Date
60	2005	7	3225769	2005-07
62	2005	8	3195866	2005-08
90	2005	9	2740553	2005-09
117	2005	10	2770715	2005-10
84	2005	11	2617333	2005-11
...
109	2015	11	4013814	2015-11
82	2015	12	4129052	2015-12
113	2016	1	3748529	2016-01
85	2016	2	3543639	2016-02
31	2016	3	4137679	2016-03

129 rows x 4 columns

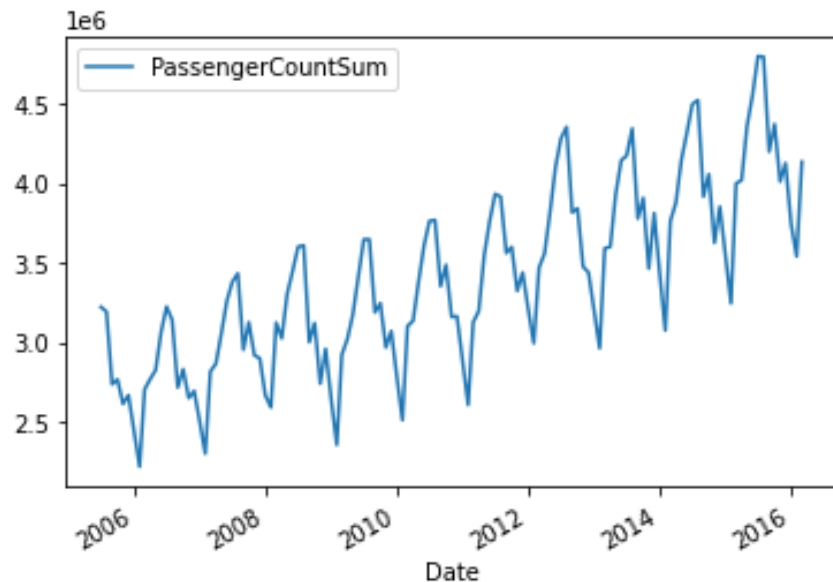

```
df_pass_by_year_pd.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 129 entries, 0 to 128  
Data columns (total 4 columns):  
#   Column                Non-Null Count  Dtype  
---  ---  
0   Year                  129 non-null    int32  
1   Month                 129 non-null    Int8  
2   PassengerCountSum     129 non-null    int64  
3   Date                  129 non-null    datetime64[ns]  
dtypes: Int8(1), datetime64[ns](1), int32(1), int64(1)  
memory usage: 2.9 KB
```

- Utilización de la librería matplotlib para la creación del gráfico

```
import matplotlib.pyplot as plt  
df_pass_by_year_pd.plot(x="Date", y="PassengerCountSum")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3663bb8760>
```



- Importación de la librería para crear posteriormente el modelo de regresión lineal

```
from sklearn import linear_model
```

- Conversión del campo Date a un ordinal:

```
import datetime as dt
df_pass_by_year_pd['Date'] = pd.to_datetime(df_pass_by_year_pd['Date'])
df_pass_by_year_pd['DateOrd'] = df_pass_by_year_pd['Date'].map(dt.datetime.toordinal)
df_pass_by_year_pd = df_pass_by_year_pd.sort_values(by=["DateOrd"])
```

	Year	Month	PassengerCountSum	Date	DateOrd
60	2005	7	3225769	2005-07-01	732128
62	2005	8	3195866	2005-08-01	732159
90	2005	9	2740553	2005-09-01	732190
117	2005	10	2770715	2005-10-01	732220
84	2005	11	2617333	2005-11-01	732251
...
109	2015	11	4013814	2015-11-01	735903
82	2015	12	4129052	2015-12-01	735933
113	2016	1	3748529	2016-01-01	735964
85	2016	2	3543639	2016-02-01	735995
31	2016	3	4137679	2016-03-01	736024

- Simplificación del DataFrame

```
df_pass_by_year_pd_simple = df_pass_by_year_pd[["PassengerCountSum", "Date", "DateOrd"]]  
df_pass_by_year_pd_simple
```

	PassengerCountSum	Date	DateOrd
60	3225769	2005-07	732128
62	3195866	2005-08	732159
90	2740553	2005-09	732190
117	2770715	2005-10	732220
84	2617333	2005-11	732251
...
109	4013814	2015-11	735903
82	4129052	2015-12	735933
113	3748529	2016-01	735964
85	3543639	2016-02	735995
31	4137679	2016-03	736024

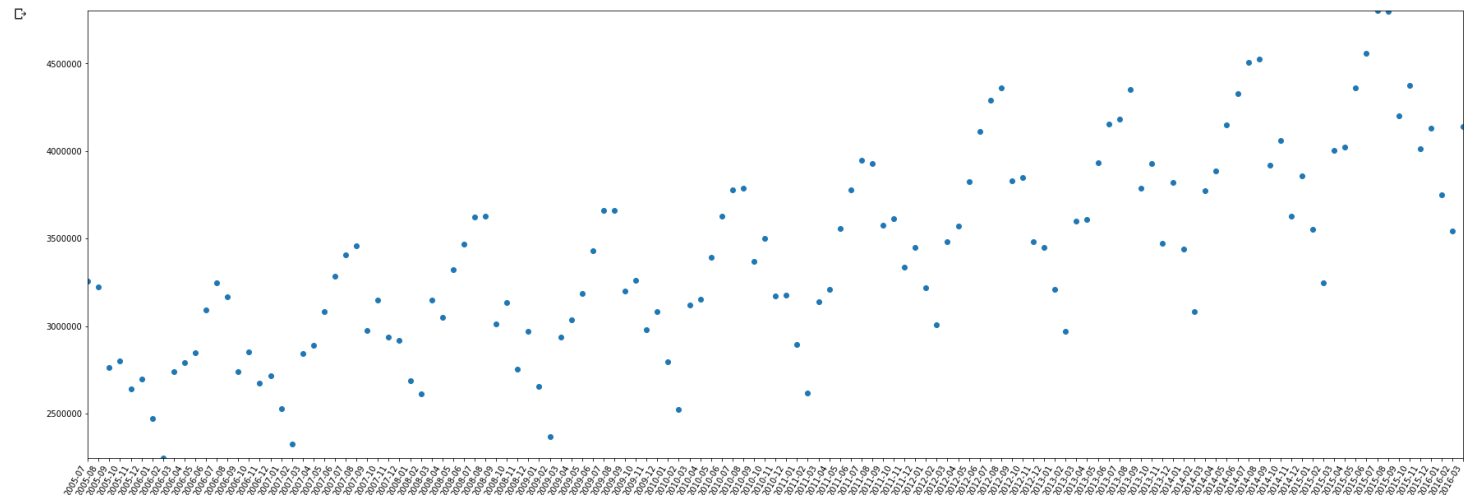
129 rows x 3 columns

- Correlación entre las variables del nuevo DataFrame:

```
[117] df_pass_by_year_pd_simple.corr().style.background_gradient(cmap='coolwarm')
```

	PassengerCountSum	DateOrd
PassengerCountSum	1.000000	0.773315
DateOrd	0.773315	1.000000

```
plt.scatter(df_pass_by_year_pd_simple["Date"], df_pass_by_year_pd_simple["AdjustedPassengerCountSum"])
plt.rcParams["figure.figsize"] = (30,10)
plt.xticks(rotation=60,ha="right")
plt.ticklabel_format(style='sci', axis='y', scilimits=(-1000000,1000000))
plt.margins(0)
plt.show()
```



- Creación la regresión:

```
model = linear_model.LinearRegression()
```

```
explicativas = df_pass_by_year_pd_simple[['DateOrd']] #independiente
objetivo = df_pass_by_year_pd_simple[['PassengerCountSum']] #dependiente
```

```
model.fit(explicativas , objetivo)
```




```
model.__dict__
```

```
{'fit_intercept': True,
 'normalize': 'deprecated',
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'n_features_in_': 1,
 'coef_': array([[385.52307302]]),
 '_residues': array([1.65561339e+13]),
 'rank_': 1,
 'singular_': array([12873.05356763]),
 'intercept_': array([-2.79601791e+08]),
 'feature_names_in_': array(['DateOrd'], dtype=object)}
```

- Creación de la predicción:

```
pred = model.predict(X=df_pass_by_year_pd_simple[['DateOrd']])
```

 `pred[:10]`

```
array([[2650445.11910808],  
       [2662396.33437181],  
       [2674347.54963553],  
       [2685913.24182624],  
       [2697864.45708996],  
       [2709430.14928061],  
       [2721381.36454433],  
       [2733332.57980806],  
       [2744127.22585267],  
       [2756078.44111639]])
```

- Ordenación de los valores en función del campo DateOrd:

```
df_pass_by_year_pd_simple.insert(3, 'Prediction', pred)  
  
pd.set_option('display.float_format', '{:.3f}'.format)  
df_pass_by_year_pd_simple = df_pass_by_year_pd_simple.sort_val-  
ues(by=["DateOrd"])  
df_pass_by_year_pd_simple
```

```
pd.set_option('display.float_format', '{:.3f}'.format)
df_pass_by_year_pd_simple = df_pass_by_year_pd_simple.sort_values('Prediction')
df_pass_by_year_pd_simple
```

	PassengerCountSum	Date	DateOrd	Prediction
60	3225769	2005-07-01	732128	2650445.119
62	3195866	2005-08-01	732159	2662396.334
90	2740553	2005-09-01	732190	2674347.550
117	2770715	2005-10-01	732220	2685913.242
84	2617333	2005-11-01	732251	2697864.457
...
109	4013814	2015-11-01	735903	4105794.720
82	4129052	2015-12-01	735933	4117360.412
113	3748529	2016-01-01	735964	4129311.627
85	3543639	2016-02-01	735995	4141262.842
31	4137679	2016-03-01	736024	4152443.012

129 rows × 4 columns

- Precisión del modelo:

```
[94] print(model.score(X=explicativas , y=objetivo))
```

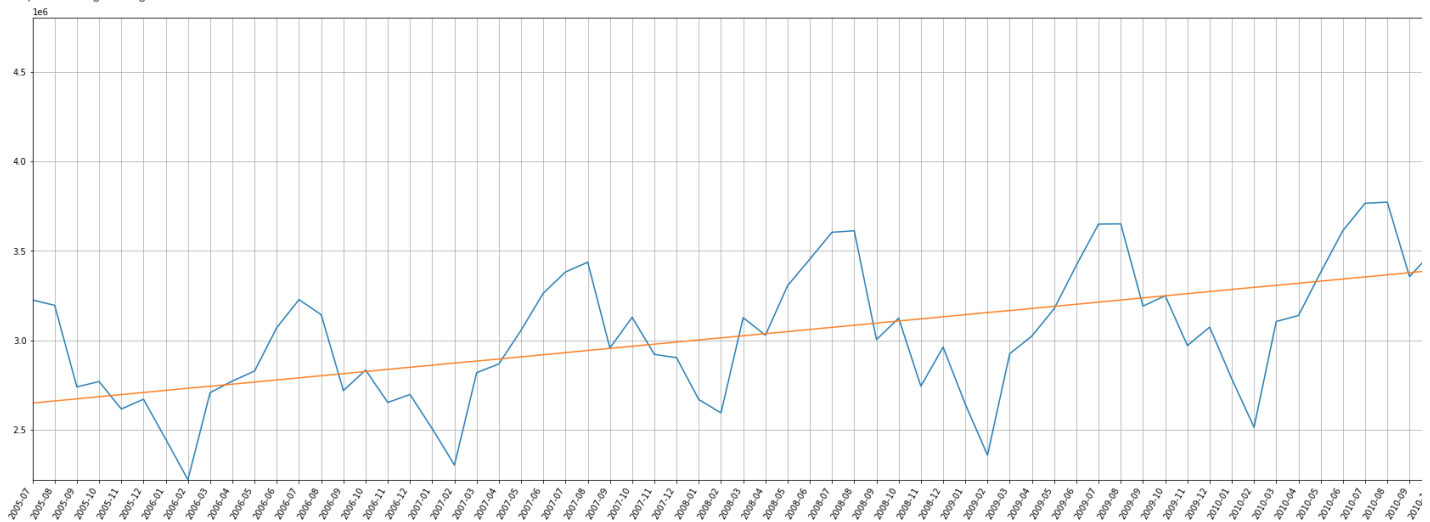
```
0.5980165190500482
```

```
df_pass_by_year_pd_simple['Date']=
pd.to_datetime(df_pass_by_year_pd['Date']).dt.strftime("%Y-%m") #esto
hace la columna un string
```

- Gráfico de regresión lineal:

```
plt.plot(df_pass_by_year_pd_simple['Date'], df_pass_by_year_pd_simple['PassengerCountSum'], label="Datos reales")
plt.plot(df_pass_by_year_pd_simple['Date'], df_pass_by_year_pd_simple['Prediction'], label="Predicción")
plt.rcParams["figure.figsize"] = (60,10)
plt.grid(True)
plt.xticks(rotation=60,ha="right")
plt.margins(0)
plt.legend()
```

<matplotlib.legend.Legend at 0x7f9a8d6ca400>



- Predicción de datos futuros:

```
dates_list = pd.date_range('2016-01-01', '2016-12-31',
                             freq='MS')
df_pred_future = pd.DataFrame(dates_list, columns=["Date"])

df_pred_future['DateOrd'] = df_pred_future['Date'].map(dt.datetime.toordinal)
df_pred_future = df_pred_future.sort_values(by=["DateOrd"])
```

```
df_pred_future = pd.DataFrame(dates_list, columns=["Date"])  
  
df_pred_future['DateOrd']=df_pred_future['Date'].map(dt.datetime.toordinal)  
df_pred_future = df_pred_future.sort_values(by=["DateOrd"])  
df_pred_future
```

	Date	DateOrd
0	2016-01-01	735964
1	2016-02-01	735995
2	2016-03-01	736024
3	2016-04-01	736055
4	2016-05-01	736085
5	2016-06-01	736116
6	2016-07-01	736146
7	2016-08-01	736177
8	2016-09-01	736208
9	2016-10-01	736238
10	2016-11-01	736269
11	2016-12-01	736299

```
pred_future = model.predict(X=df_pred_future[['DateOrd']])  
df_pred_future.insert(2, 'Prediction' , pred_future)
```

