

Importing Data/Exploratory Analysis:

AIT 614-SecDL2_Team5_Sys Python ▾ ☆
File Edit View Run Help Last edit was 58 minutes ago Provide feedback

AIT 614 - Big Data Essentials
Final Project: House Pricing Analysis
Notebook 1
Course Section #: AIT 614-DL2
Team 5: Nafisa Ahmed, Peter Bishay, Charles Gilbertson, Sneha Kumaran, Cora Sula

Data Cleansing/Exploratory Analysis

Cmd 2

```
1 # import dataset
2 housing_data = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/nahmed27@gmu.edu/DC_Properties.csv", inferSchema = "true") #added inferSchema to produce the ML algorithm
3
4 # display the first few rows of the dataset
5 display(housing_data)
```

(3) Spark Jobs

housing_data: pyspark.sql.dataframe.DataFrame = [c0: integer, BATHRM: integer ... 47 more fields]

Table +

c0	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	SALE_NUM	GBA	
1	0	4	0	Warm Cool	Y	2	8	4	1910	1988	1972	3	2003-11-25T00:00:00Z	1095000	Q	1	2522
2	1	3	1	Warm Cool	Y	2	11	5	1898	2007	1972	3	2000-08-17T00:00:00Z	null	U	1	2567
3	2	3	1	Hot Water Rad	Y	2	9	5	1910	2009	1984	3	2016-06-21T00:00:00Z	2100000	Q	3	2522
4	3	3	1	Hot Water Rad	Y	2	8	5	1900	2006	1984	3	2006-07-12T00:00:00Z	1602000	Q	1	2484
5	4	2	1	Warm Cool	Y	1	11	3	1913	2012	1985	3	null	null	U	1	5255
6	5	3	2	Hot Water Rad	Y	1	10	5	1913	null	1972	4	2010-02-26T00:00:00Z	1950000	Q	1	5344
7	6	1	0	Warm Cool	Y	2	5	2	1917	1988	1957	2	2011-05-02T00:00:00Z	null	U	1	1260
8	7	2	1	Warm Cool	V	9	6	1	1904	2011	1973	2	2011-06-10T00:00:00Z	1100000	P	1	1000

4,566 rows | Truncated data | 28.44 seconds runtime

Refreshed 57 minutes ago

dataricks

AIT 614-SecDL2_Team5_Sys Python ▾ ☆
File Edit View Run Help Last edit was 1 hour ago Provide feedback

Cmd 4

```
1 # Display the total rows of the dataset
2 housing_data.count()
```

(2) Spark Jobs

158957

Command took 7.77 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 5

```
1 # display summary/descriptive statistics
2 display(housing_data.summary())
```

(2) Spark Jobs

Table +

summary	c0	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	SALE_NUM	GBA
count	158957	158957	158957	158957	158957	106996	158957	158957	158957	158957	158957	158957	158957	158957	158957	158957	
mean	7947.80	1.8106783595356358	0.04582371333127827	null	0.0	1.1980392891954712	6.187736306045031	2.7325062752819944	1941.987579244546	1988.2435374654012	1963.7180243713708	1.					
stddev	45887.07770712796	0.9763959589356358	0.5875714744690714	null	0.0	0.5969244151364597	2.618164876297589	1.3588642440199536	33.6402337602978	16.575785685001474	24.923150118202688	2.					
min	0	0	0	Air Exchng	0	0.0	0	0	1754.0	20.0	1800	0.					
25%	39733	1	0	null	0.0	1.0	4	2	1918.0	1985.0	1954	2.					
50%	79465	2	0	null	0.0	1.0	6	3	1937.0	2004.0	1963	2.					
75%	119222	2	1	null	0.0	1.0	7	3	1960.0	2010.0	1975	3.					

8 rows | 1:17 minutes runtime

Command took 1.17 minutes -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 6

```
1 # display the data schema
2 housing_data.printSchema()
```

root
|-- c0: integer (nullable = true)
|-- BATHRM: integer (nullable = true)
|-- HF_BATHRM: integer (nullable = true)
|-- HEAT: string (nullable = true)
|-- AC: string (nullable = true)

AIT 614-SecDL2_Team5_Sys Python

```

1 # drop unnecessary columns
2 column_to_drop = ["CENSUS_BLOCK", "SQUARE", "X", "Y", "CHPLX_NUM", "USECODE",
3 "GIS_LAST_MDO_DTM"]
4 housing_data = housing_data.drop(*column_to_drop)

# housing_data: pyspark.sql_dataframe.DataFrame
# BATHRM: integer
# HF_BATHRM: integer
# HEAT: string
# AC: string
# NUM_UNITS: double
# ROOMS: integer
# BEDRM: integer
# BDRML: integer
# AYB: double
# YR_RMDL: double
# EYB: integer
# STORIES: double
# SALEDATE: timestamp
# PRICE: double
# QUALIFIED: string
# GBA: double
# BLDG_NUM: integer
# STYLE: string
# STRUCTT: string
# SQFT: string
# CNTL: string
# EXTRALL: string
# ROOF: string

```

Command took 0.11 seconds -- by naheed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 8

```

1 # check for NULL values
2 from pyspark.sql.functions import col, count, when
3 null_counts = housing_data.select((count(when(col(c).isNull(), c)).alias(c) for c in housing_data.columns))
4
5 # display the null value counts
6 display(null_counts)

```

(2) Spark Jobs

null_counts: pyspark.sql_dataframe.DataFrame = [BATHRM: long, HF_BATHRM: long ... 28 more fields]

Table

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDG_NUM	STYLE	STF
1	0	0	0	52261	0	0	271	78029	0	52305	26770	60741	0	52261	0	52261	522

1 row | 7.79 seconds runtime

Refreshed 1 hour ago

Command took 0.79 seconds -- by naheed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 9

```

1 # drop the NULL values for the 'PRICE' variable
2 housing_data = housing_data.dropna(subset=['PRICE'])

```

housing_data: pyspark.sql_dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 28 more fields]

Command took 0.89 seconds -- by naheed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 10

```

1 # display total rows for the data
2 housing_data.count()

```

(2) Spark Jobs

98216

Command took 0.71 seconds -- by naheed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

AIT 614-SecDL2_Team5_Sys Python ▾ ☆

Last edit was 1 hour ago Provide feedback

Cmd 11

```

1 # check for NULL Values again
2 null_counts = housing_data.select([count(when(col(c).isNull(), c)).alias(c) for c in housing_data.columns])
3
4 # Display the null counts row
5 display(null_counts)

> (2) Spark Jobs
> null_counts: pyspark.sql.dataframe.DataFrame = [BATHRM: long, HF_BATHRM: long ... 28 more fields]

Table v +

```

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDG_NUM	STYLE	STI
1	0	0	0	40316	0	0	112	40342	0	40349	1	0	0	40316	0	40316	403

↓ 1 row | 6.18 seconds runtime

Refreshed 1 hour ago

Command took 6.18 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 12

```

1 # display the data schema
2 housing_data.printSchema()

root
|- BATHRM: integer (nullable = true)
|- HF_BATHRM: integer (nullable = true)
|- HEAT: string (nullable = true)
|- AC: string (nullable = true)
|- NUM_UNITS: double (nullable = true)
|- ROOMS: Integer (nullable = true)
|- BEDRM: Integer (nullable = true)
|- AYB: double (nullable = true)
|- YR_RMDL: double (nullable = true)
|- EYB: Integer (nullable = true)
|- STORIES: double (nullable = true)
|- SALEDATE: timestamp (nullable = true)
|- PRICE: double (nullable = true)
|- QUALIFIED: string (nullable = true)
|- GBA: double (nullable = true)
|- BLDG_NUM: integer (nullable = true)
|- STYLE: string (nullable = true)

```

AIT 614-SecDL2_Team5_Sys Python ▾ ☆

Last edit was 1 hour ago Provide feedback

Cmd 13

```

1 # display the first few rows of the dataset
2 display(housing_data)

> (1) Spark Jobs

```

Table v +

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDG_NUM	STYLE	
1	4	0	Warm Cool	Y	2	8	4	1910	1988	1972	3	2003-11-25T00:00:00Z	1095000	Q	2522	1	3 Stor
2	3	1	Hot Water Rad	Y	2	9	5	1910	2009	1984	3	2016-06-21T00:00:00Z	2100000	Q	2522	1	3 Stor
3	3	1	Hot Water Rad	Y	2	8	5	1900	2003	1984	3	2006-07-12T00:00:00Z	1602000	Q	2484	1	3 Stor
4	3	2	Hot Water Rad	Y	1	10	5	1913	null	1972	4	2010-02-26T00:00:00Z	1950000	Q	5344	1	4 Stor
5	3	1	Hot Water Rad	Y	2	8	4	1906	2011	1972	3	2011-09-29T00:00:00Z	1050000	Q	2401	1	3 Stor
6	3	1	Warm Cool	Y	2	7	3	1908	2008	1967	2	2016-05-03T00:00:00Z	1430000	Q	1488	1	2 Stor
7	3	1	Warm Cool	Y	2	5	3	1917	2000	1967	2	2011-09-30T00:00:00Z	1325000	Q	2692	1	2 Stor

↓ 9,064 rows | Truncated data | 2.08 seconds runtime

Refreshed 1 hour ago

Command took 2.08 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 14

Queries

Cmd 15

```

1 # Create a temporary view of the housing_data dataframe
2
3 housing_data.createOrReplaceTempView("dc_properties")

Command took 0.34 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

```

Cmd 16

```

1 %sql
2 select * from dc_properties

> (1) Spark Jobs
> dc_properties: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer, ... 28 more fields]

```

Queries:

databricks

AIT 614-SecDL2_Team5_Sys Python ⭐

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Run all Cluster Share Publish

Queries

Cmd 15

```
1 # Create a temporary view of the housing_data dataframe
2
3 housing_data.createOrReplaceTempView("dc_properties")
```

Command took 0.34 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 16

```
1 %sql
2 select * from dc_properties
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 28 more fields]

Table +

	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	\$
1	4	0	Warm Cool	Y	2	8	4	1910	1988	1972	3	2
2	3	1	Hot Water Rad	Y	2	9	5	1910	2009	1984	3	2
3	3	1	Hot Water Rad	Y	2	8	5	1900	2003	1984	3	2
4	3	2	Hot Water Rad	Y	1	10	5	1913	null	1972	4	2
5	3	1	Hot Water Rad	Y	2	8	4	1906	2011	1972	3	2
6	3	1	Warm Cool	Y	2	7	3	1908	2008	1967	2	2

databricks

AIT 614-SecDL2_Team5_Sys Python ⭐

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Run all Cluster Share Publish

Queries

Cmd 16

```
1 # Create a temporary view of the housing_data dataframe
2
3 housing_data.createOrReplaceTempView("dc_properties")
```

Command took 0.34 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 16

```
1 %sql
2 select * from dc_properties
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 28 more fields]

Table +

	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	\$
1	4	0	Warm Cool	Y	2	8	4	1910	1988	1972	3	2
2	3	1	Hot Water Rad	Y	2	9	5	1910	2009	1984	3	2
3	3	1	Hot Water Rad	Y	2	8	5	1900	2003	1984	3	2
4	3	2	Hot Water Rad	Y	1	10	5	1913	null	1972	4	2
5	3	1	Hot Water Rad	Y	2	8	4	1906	2011	1972	3	2
6	3	1	Warm Cool	Y	2	7	3	1908	2008	1967	2	2

↓ ↓ 9,064 rows | Truncated data | 2.22 seconds runtime

Refreshed 1 hour ago

This result is stored as PySpark data frame _sqldf and in the IPython output cache as Out[13]. Learn more

Command took 2.22 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 17

databricks

AIT 614-SecDL2_Team5_Sys Python

File Edit View Run Help Last edit was 1 hour ago Provide feedback

N Run all Cluster Share Publish

Cmd 17

This result is stored as PySpark data frame `_sqldf` and in the IPython output cache as `Out[13]`. Learn more
Command took 2.22 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

```
1 %sql
2 SELECT AVG(PRICE) AS average_price FROM dc_properties;
```

(2) Spark Jobs

`_sqldf: pyparck.sql.DataFrame = [average_price: double]`

Table +

average_price
931351.5949336156

1 row | 3.48 seconds runtime Refreshed 1 hour ago

This result is stored as PySpark data frame `_sqldf` and in the IPython output cache as `Out[14]`. Learn more
Command took 3.48 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 18

```
1 %sql
2 SELECT QUADRANT, AVG(PRICE) AS average_price
3 FROM dc_properties
4 WHERE PRICE IS NOT NULL AND QUADRANT IS NOT NULL
5 GROUP BY QUADRANT;
```

(2) Spark Jobs

SQL

QUADRANT	average_price
NW	1317785.1587480297
SW	347048.9480189022
NE	392871.5477633875
SE	368226.04414454184

4 rows | 4.97 seconds runtime Refreshed 1 hour ago

databricks

AIT 614-SecDL2_Team5_Sys Python

File Edit View Run Help Last edit was 1 hour ago Provide feedback

N Run all Cluster Share Publish

Cmd 18

This result is stored as PySpark data frame `_sqldf` and in the IPython output cache as `Out[15]`. Learn more
Command took 4.97 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

```
1 %sql
2 SELECT QUADRANT, STRUCT1 AVG(PRICE) AS average_price
3 FROM dc_properties
```

Table +

QUADRANT	average_price
NW	1317785.1587480297
SW	347048.9480189022
NE	392871.5477633875
SE	368226.04414454184

4 rows | 4.97 seconds runtime Refreshed 1 hour ago

This result is stored as PySpark data frame `_sqldf` and in the IPython output cache as `Out[15]`. Learn more
Command took 4.97 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 19

```
1 %sql
2 SELECT QUADRANT, STRUCT1 AVG(PRICE) AS average_price
3 FROM dc_properties
```

Cmd 19

```

1  %sql
2  SELECT QUADRANT, STRUCTA AVG(PRICE) AS average_price
3  FROM dc_properties
4  WHERE PRICE IS NOT NULL AND QUADRANT IS NOT NULL AND STRUCT IS NOT NULL
5  GROUP BY QUADRANT, STRUCT;
6

```

(2) Spark Jobs

_sqldf: pypark.sql.dataframe.DataFrame = [QUADRANT: string, STRUCT: string ... 1 more field]

QUADRANT	STRUCT	average_price
1 NW	Semi-Detached	664421.4532780708
2 NW	Row Inside	650918.9568155885
3 SE	Multi	344440.0215568862
4 NE	Town Inside	463169.3214285714
5 SW	Town End	317879.0909090909
6 NW	Town End	387005

30 rows | 4.21 seconds runtime

This result is stored as PySpark data frame _sqldf and in the IPython output cache as Out[16]. Learn more

Command took 4.21 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 20

Visualizations:

Cmd 20

Visualizations

Cmd 21

```

1  # convert spark data frame to a pandas data frame to create visualizations
2  housing_data_pd = housing_data.toPandas()
3
4  #display pandas data
5  housing_data_pd.head()

```

(1) Spark Jobs

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDG_NUM	STYLE	STRUCT	GRADE	CNTDN	EXTWALL	ROOF	INTWALL	KITCHENS	FIREPLACES	LANDAREA	SOURCE	LIVING_GBA		
0	4	0	Warm	Cool	Y	2.0	8	4	1910.0	1988.0	1972	3.0	2003-11-25	1085000.0	Q	2522.0	1	3 Story	Row Inside	Very Good	Common Brick	Metal-Sms	Hardwood	2.0	5	1600	Residential	NaN	
1	3	1	Hot	Water	Rad	2.0	9	5	1910.0	2008.0	1984	3.0	2016-06-21	2100000.0	Q	2522.0	1	3 Story	Row Inside	Very Good	Common Brick	Built Up	Hardwood	2.0	4	1600	Residential	NaN	
2	3	1	Hot	Water	Rad	2.0	8	5	1900.0	2003.0	1984	3.0	2006-07-12	1602000.0	Q	2484.0	1	3 Story	Row Inside	Very Good	Common Brick	Built Up	Hardwood	2.0	3	1600	Residential	NaN	
3	3	2	Hot	Water	Rad	1.0	10	5	1913.0	NaN	1972	4.0	2010-02-26	1950000.0	Q	5344.0	1	4 Story	Row Inside	Very Good	Common Brick	Built Up	Hardwood	1.0	4	2196	Residential	NaN	
4	3	1	Hot	Water	Rad	2.0	8	4	1906.0	2011.0	1972	3.0	2011-09-29	1050000.0	Q	2401.0	1	3 Story	Row Inside	Very Good	Average	Common Brick	Metal-Sms	Hardwood	2.0	1	1627	Residential	NaN

Command took 8.03 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 22

```

1  #Correlation matrix
2  housing_data_pd.corr()

```

BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	PRICE	GBA	BLDG_NUM	KITCHENS	FIREPLACES	LANDAREA	LIVING_GBA	
BATHRM	1.000000	0.204066	0.353612	0.710764	0.709358	-0.071179	0.246807	0.245529	0.023327	-0.005536	0.696760	-0.035512	0.401115	-0.002470	0.440420	0.722723
HF_BATHRM	0.294066	1.000000	-0.177140	0.394385	0.415388	-0.045465	0.167185	0.231376	0.032569	-0.013113	0.257084	-0.006251	-0.154734	-0.002228	0.323970	0.481738
NUM_UNITS	0.353612	-0.177140	1.000000	0.537695	0.340115	-0.111175	-0.030150	-0.124966	0.009943	-0.005469	0.206374	-0.019469	0.911839	-0.041616	-0.056729	NaN

AIT 614-SecDL2_Team5.Sys Python

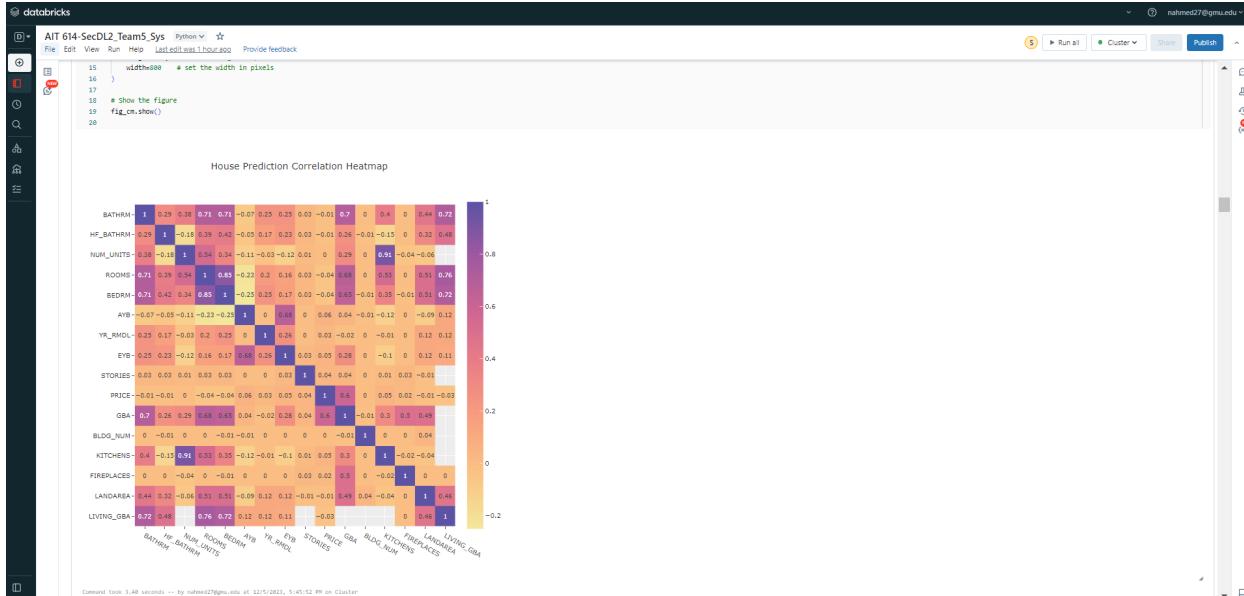
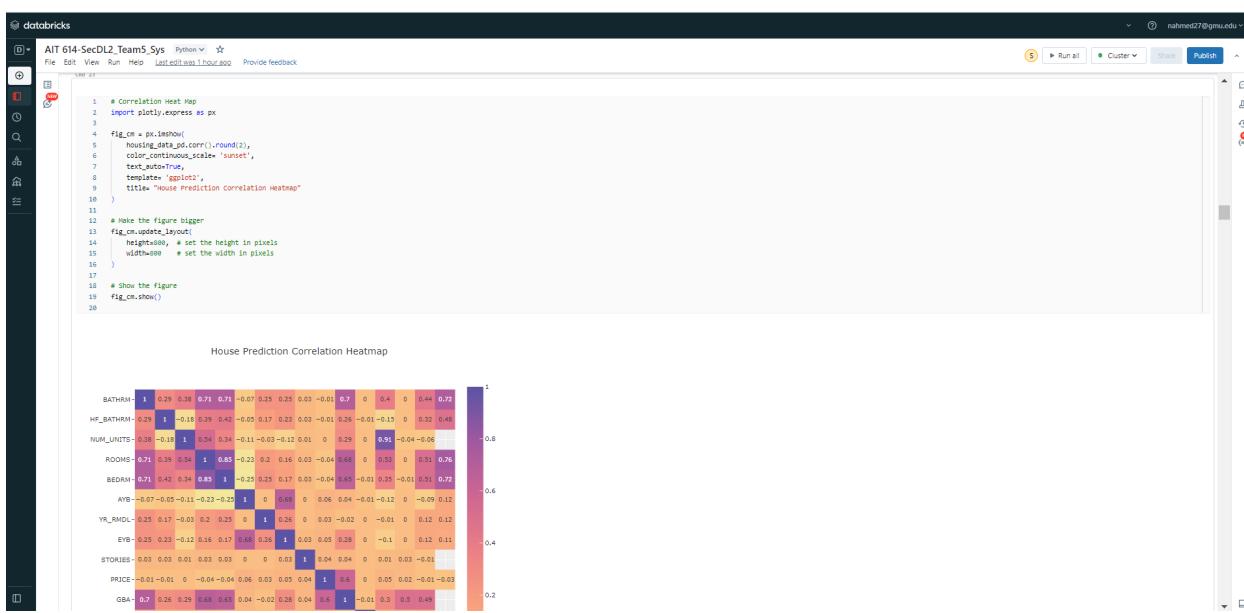
```

1 #Correlation matrix
2 housing_data_pd.corr()

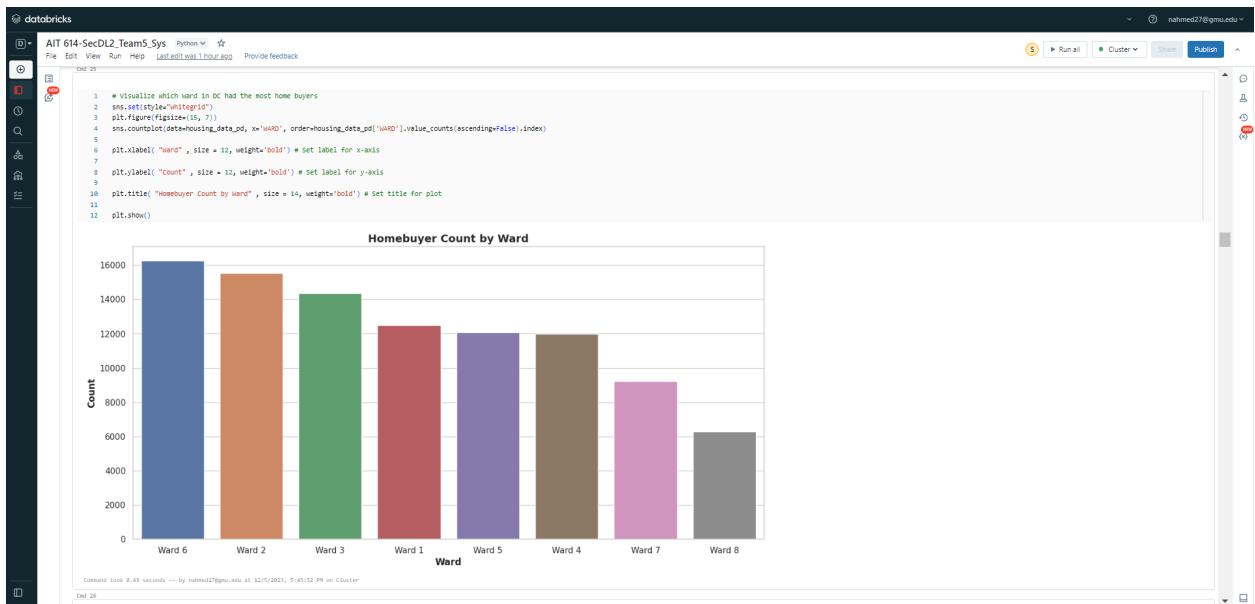
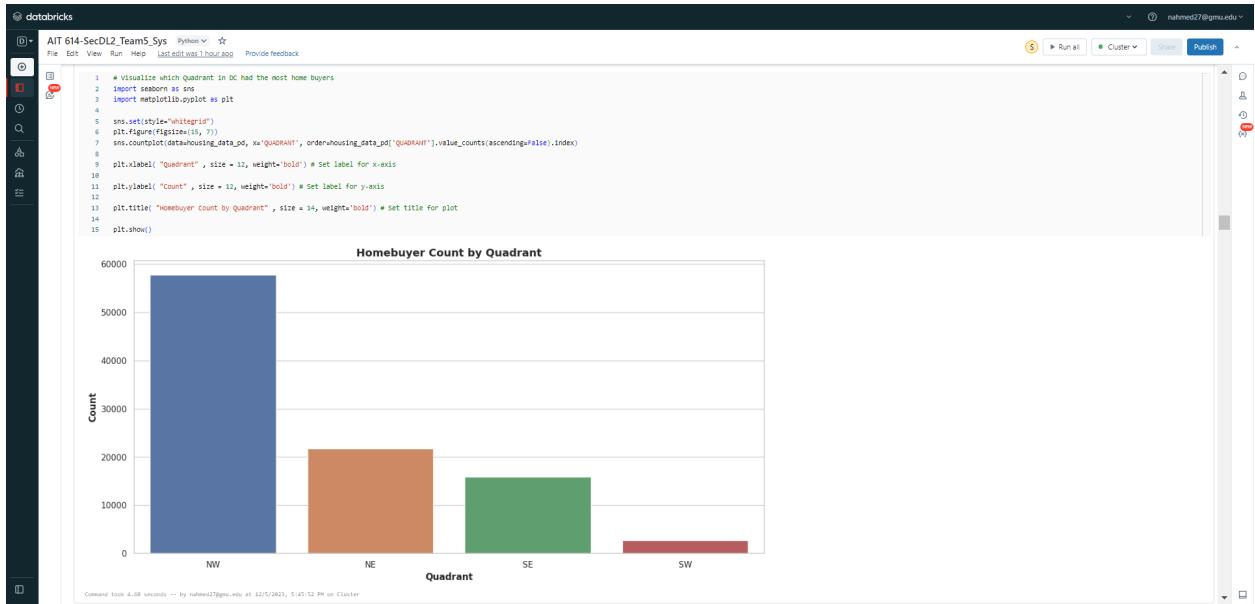
```

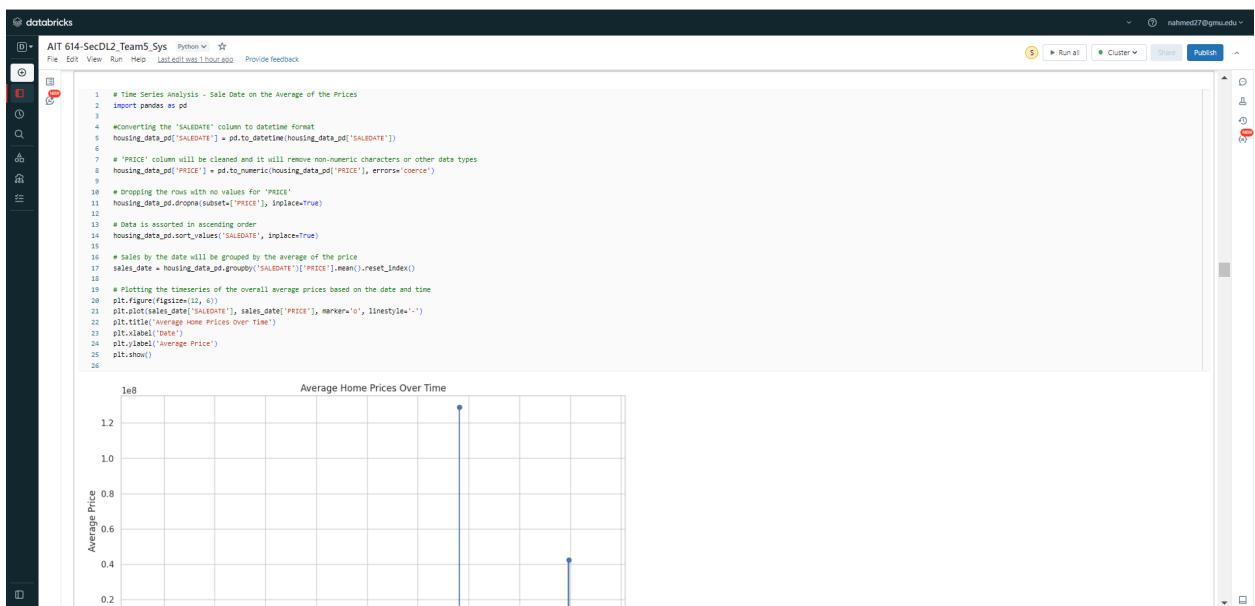
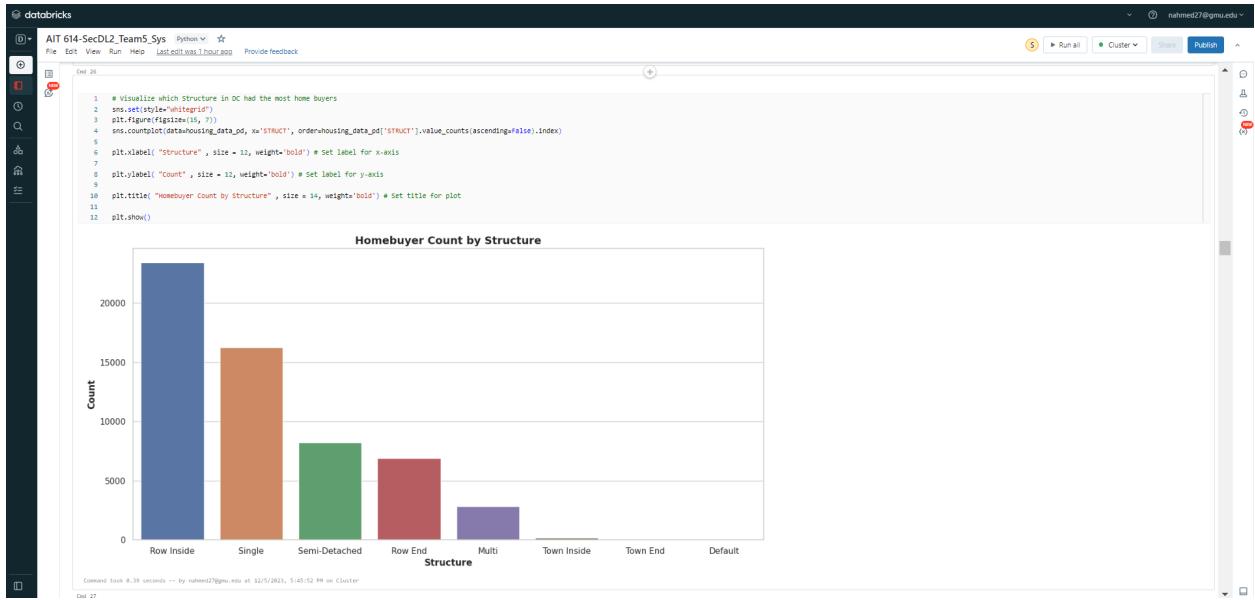
	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	PRICE	GBA	BLDG_NUM	KITCHENS	FIREPLACES	LANDAREA	LIVING_GBA
BATHRM	1.000000	0.254088	0.385612	0.710784	0.709558	-0.071179	0.246087	0.245529	0.023327	-0.005358	0.696760	-0.03532	0.40115	-0.02470	0.440420	0.722723
HF_BATHRM	0.254088	1.000000	-0.177140	0.394385	0.415388	-0.045465	0.167165	0.231378	0.025598	-0.013113	0.257704	-0.006251	-0.154734	-0.022228	0.323970	0.451736
NUM_UNITS	0.385612	-0.177140	1.000000	0.537695	0.340115	-0.111175	-0.301051	-0.249684	0.009943	0.285374	0.019489	0.918139	-0.041618	-0.056729	NAN	
ROOMS	0.710784	0.394385	0.537695	1.000000	0.447760	-0.253228	0.202394	0.146280	0.025785	-0.032341	0.650518	-0.048419	0.528590	-0.044488	0.514334	0.759645
BEDRM	0.709558	0.415388	0.340115	0.447760	1.000000	-0.248550	0.253182	0.170752	0.028312	0.043331	0.645725	-0.07643	0.352900	-0.05711	0.514365	0.717704
AYB	-0.071179	-0.045465	-0.111175	-0.253228	-0.248550	1.000000	0.033348	0.025803	0.003381	0.030272	-0.019683	-0.00731	-0.013625	0.011320	0.110223	0.124243
YR_RMDL	0.246087	0.167165	-0.030150	0.202394	0.253182	0.033348	1.000000	0.025803	0.003381	0.030272	-0.019683	-0.00731	-0.011674	0.010563		
EYB	0.245529	0.231378	-0.124966	0.164288	0.170732	0.043481	0.256093	1.000000	0.023950	0.049627	0.284696	0.000756	-0.102144	-0.000198	0.116174	
STORIES	0.023327	0.032569	0.009943	0.025766	0.026312	0.004003	0.003365	0.039550	1.000000	0.042337	0.042322	-0.00702	0.011565	0.027901	-0.014466	NAN
PRICE	-0.066536	-0.015113	-0.004689	-0.032421	-0.043331	0.056041	0.030072	0.049627	0.042337	1.000000	0.005078	0.003648	0.045213	0.023969	-0.005457	-0.027106
GBA	0.696760	0.257704	0.206374	0.206374	0.206374	0.045725	0.055862	0.019683	0.054041	0.035662	1.000000	-0.009831	0.299863	0.526994	0.494546	NAN
BLDG_NUM	-0.003512	-0.062551	-0.001949	-0.04819	-0.007443	-0.070578	-0.007031	0.000756	-0.00702	0.003948	-0.009831	1.000000	-0.02541	-0.000445	0.040238	NAN
KITCHENS	0.40115	-0.14734	0.911639	0.528950	0.532900	-0.11075	-0.013630	-0.012147	0.011568	0.045213	0.296983	-0.02541	1.000000	-0.015232	-0.03189	NAN
FIREPLACES	-0.002470	-0.022228	-0.041616	-0.044489	-0.05711	0.01507	0.001938	-0.000198	0.027901	0.023089	0.502894	-0.000448	-0.015238	1.000000	-0.001991	-0.004179
LANDAREA	0.440420	0.323970	0.514334	0.514355	0.514355	0.110823	0.116174	-0.014468	-0.005457	0.494546	0.040238	-0.039169	-0.001991	1.000000	0.455107	
LIVING_GBA	0.722723	0.401736	Nan	0.759645	0.717004	0.115434	0.122423	0.105653	Nan	0.027106	Nan	Nan	Nan	-0.004179	0.455107	1.000000

Command took 8.19 seconds -- by nahmed27@gsu.edu at 12/5/2021, 5:43:52 PM on Cluster



Command took 3.48 seconds -- by nahmed27@gsu.edu at 12/5/2021, 5:45:12 PM on Cluster



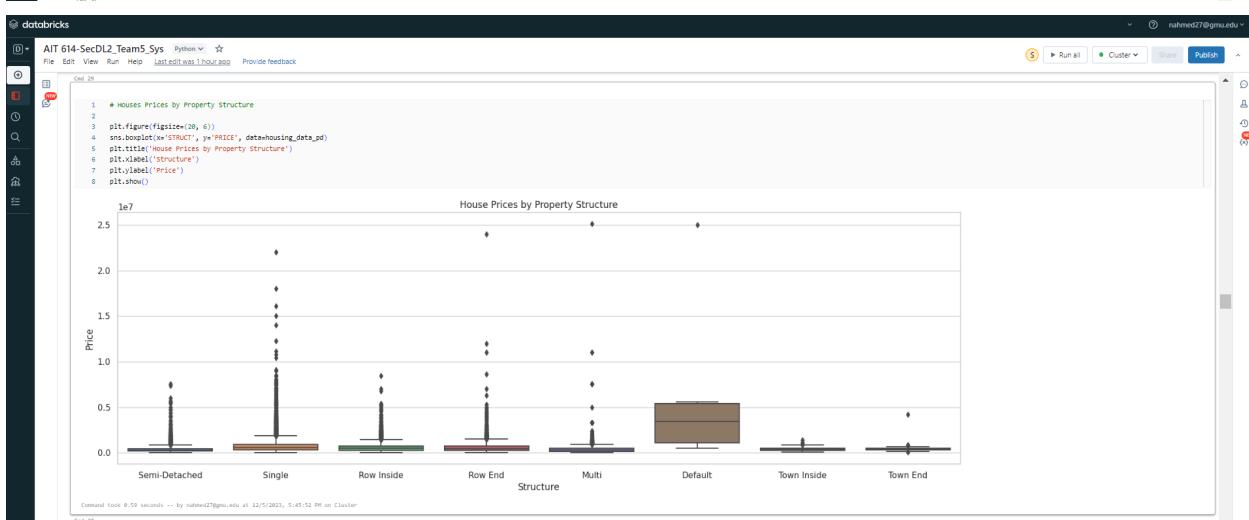
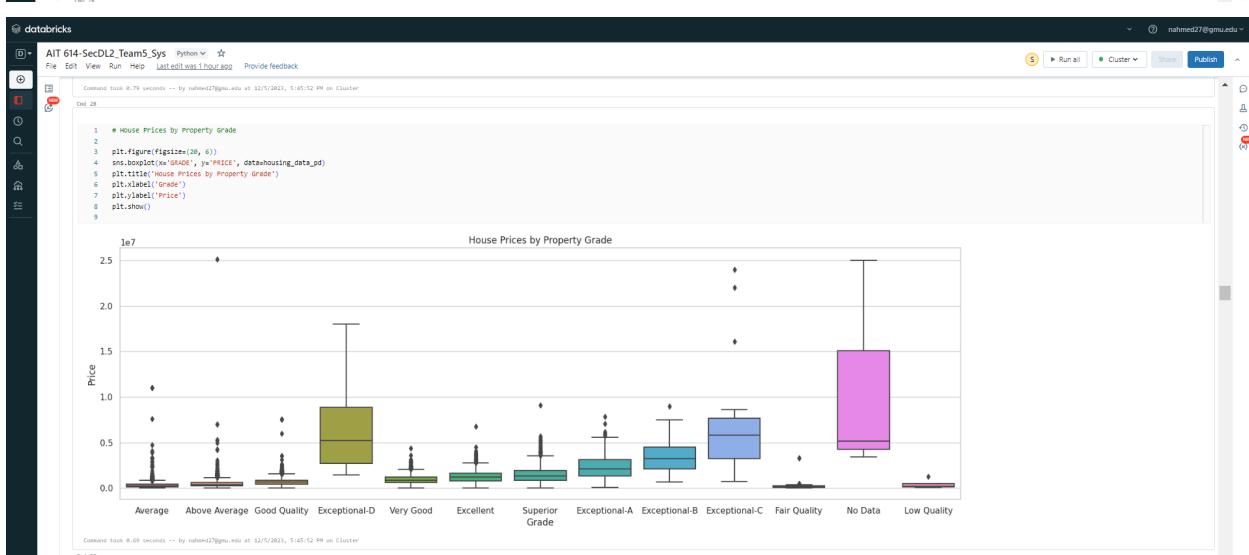


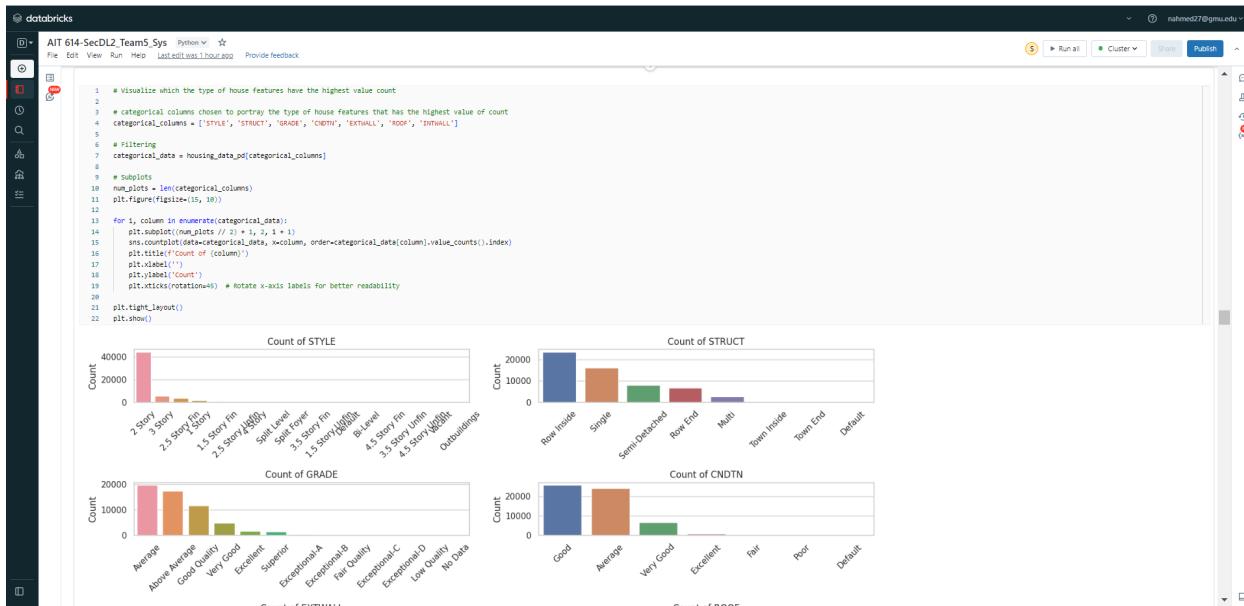
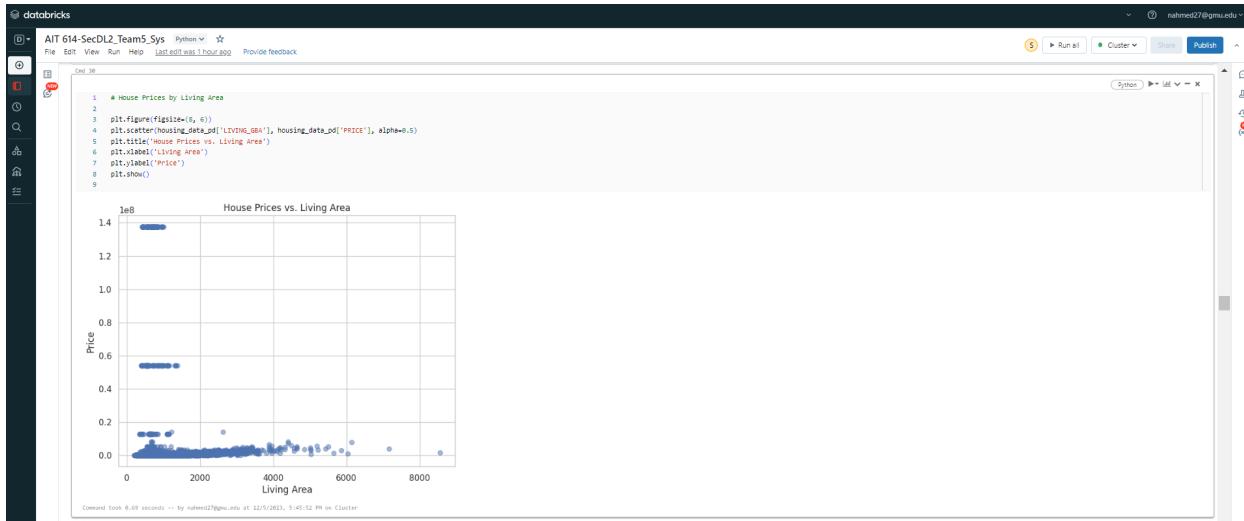
AIT 614-SecDL2_Team5.Sys Python

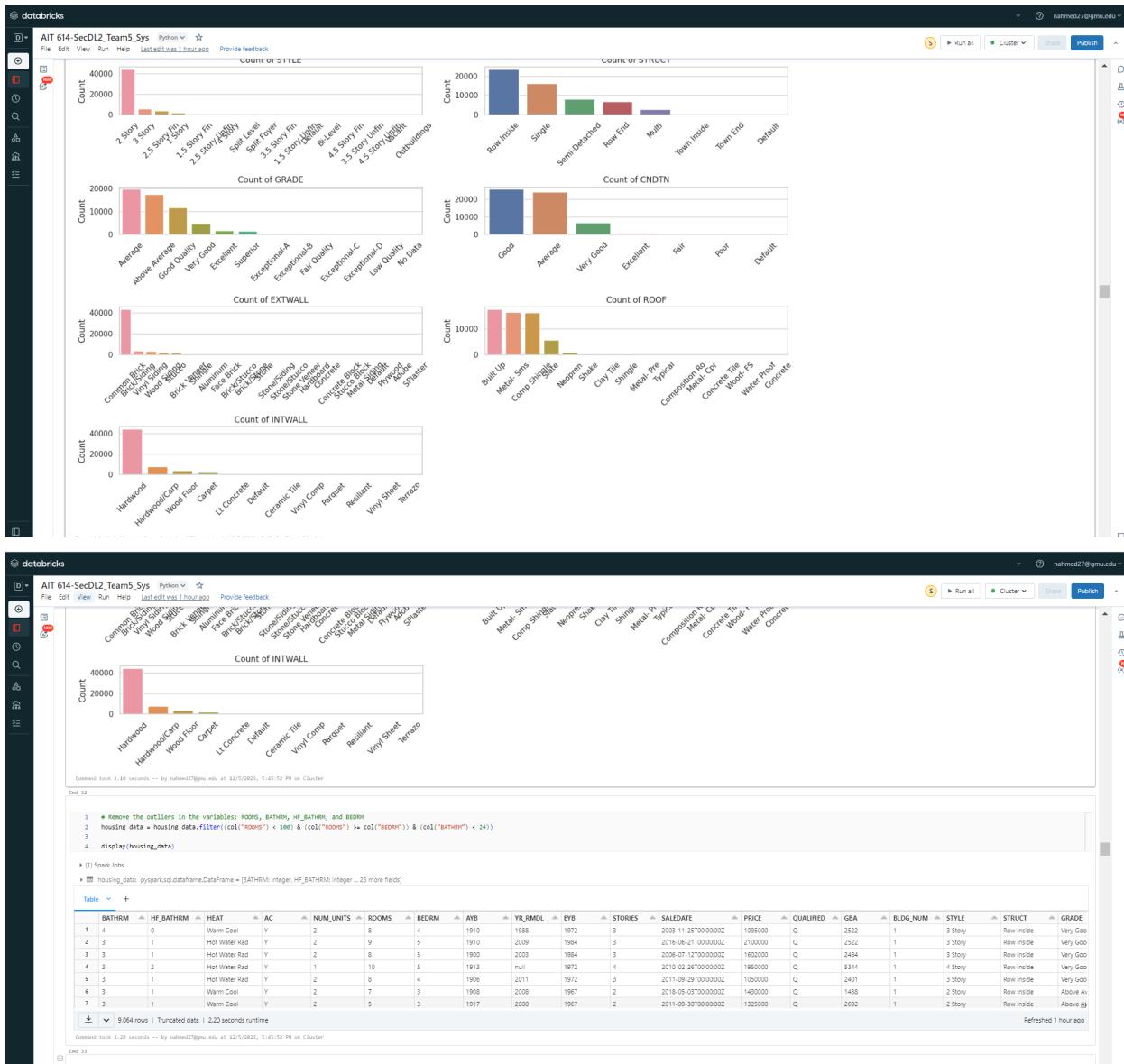
```

1 # Importing the required libraries
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 # Reading the CSV file into a Pandas DataFrame
7 housing_data_pd = pd.read_csv('housing_data.csv')
8
9 # Dropping rows with missing values for 'PRICE'
10 housing_data_pd.dropna(subset=['PRICE'], inplace=True)
11
12 # Data is sorted in ascending order
13 housing_data_pd.sort_values(['SALEDATE'], inplace=True)
14
15 # Sales by the date will be grouped by the average of the price
16 sales_date = housing_data_pd.groupby(['SALEDATE'])['PRICE'].mean().reset_index()
17
18 # Plotting the timeseries of the overall average prices based on the date and time
19 plt.figure(figsize=(12, 4))
20 plt.plot(sales_date['SALEDATE'], sales_date['PRICE'], marker='o', linestyle='.')
21 plt.title('Average Home Prices Over Time')
22 plt.xlabel('Date')
23 plt.ylabel('Average Price')
24 plt.show()
25
26

```







Feature Preprocessing/Training & Test Data Split:

The screenshot shows a Databricks notebook titled "Feature Preprocessing/Training & Test Data Split". The notebook contains Python code for data preprocessing, specifically for a housing dataset. The code includes imports for StringIndexer, OneHotEncoder, VectorAssembler, and Imputer from pyspark.ml. It defines categorical and numeric columns, creates lists of StringIndexers and OneHotEncoders, and assembles all features into a single vector. The label is defined as 'PRICE'. A pipeline is created with stages including indexers, encoders, imputer, and assembler. The pipeline is then fitted to the data, and the transformed data is shown.

```
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler, Imputer
from pyspark.ml import Pipeline

# Define categorical columns
categorical_cols = ['HEAT', 'AC', 'QUALIFIED', 'STYLE', 'STRUCT', 'GRADE', 'ONDIN', 'EXTWALL', 'ROOF', 'INTWALL', 'SOURCE', 'WARD', 'QUADRANT']

# Create a list of StringIndexes for categorical columns
indexers = [StringIndexer(inputCol=col, outputCol=col + "_index", handleInvalid="keep") for col in categorical_cols]

# Create a list of OnehotEncoders for indexed categorical columns
encoders = [OneHotEncoder(inputCol=col + "_index", outputCol=col + "_encoded") for col in categorical_cols]

# Define the numeric columns
numeric_cols = ['BATHRM', 'HF_BATHRM', 'NUM_UNITS', 'ROOMS', 'BEDRM', 'AYB', 'VR_RMVL', 'EYB', 'STORIES',
                'GBA', 'BLDG_NUM', 'KITCHENS', 'FIREPLACES', 'LANDAREA', 'LIVING_GBA']

# Impute missing values in numeric columns using the mean
imputer = Imputer(inputCols=numeric_cols, outputCols=[col + "_imputed" for col in numeric_cols], strategy="mean")

# Assemble all features into a single vector
assembler = VectorAssembler(inputCols=[col + "_imputed" for col in numeric_cols] + [col + "_encoded" for col in categorical_cols],
                            outputCol="features")

# Define the label column
label = "PRICE"

# Create a pipeline
pipeline = Pipeline(stages=indexers + encoders + [imputer, assembler])

# Fit the pipeline to the data
pipeline_model = pipeline.fit(housing_data)

# Transform the data
transformed_data = pipeline_model.transform(housing_data)

# Show the transformed data
transformed_data.select("features", label).show()
```

Log: Run 1 run to an experiment in MLflow. Learn more

transformed_data: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 70 more fields]

data bricks

AI 614-SecDL2_Team5.Sys Python

File Edit View run Help Last edit date: 12/5/2023 5:45:52 PM Provide feedback

33 # Transform the data
34 transformed_data = pipeline_model.transform(housing_data)
35
36 # Show the transformed data
37 transformed_data.select('features', 'label').show()

(75) Spark Jobs

(1) MLflow run
Logged 1 run to an experiment in MLflow. Learn more

transformed_data: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 70 more fields]

	features	PRICE
[146]	[0,2,3,4,5,6,...]	[1095000.0]
[146]	[0,1,2,3,4,5,...]	[2180000.0]
[146]	[0,1,2,3,4,5,...]	[1682000.0]
[146]	[0,1,2,3,4,5,...]	[1580000.0]
[146]	[0,1,2,3,4,5,...]	[1520000.0]
[146]	[0,1,2,3,4,5,...]	[1320000.0]
[146]	[0,1,2,3,4,5,...]	[1325000.0]
[146]	[0,1,2,3,4,5,...]	[1240000.0]
[146]	[0,1,2,3,4,5,...]	[592250.0]
[146]	[0,1,2,3,4,5,...]	[33252.0]
[146]	[0,2,3,4,5,6,...]	[1095000.0]
[146]	[0,1,2,3,4,5,...]	[1065000.0]
[146]	[0,1,2,3,4,5,...]	[1108000.0]
[146]	[0,2,3,4,5,6,...]	[20000.0]
[146]	[0,1,2,3,4,5,...]	[10000.0]
[146]	[0,1,2,3,4,5,...]	[12120000.0]
[146]	[0,1,2,3,4,5,...]	[139500.0]
[146]	[0,1,2,3,4,5,...]	[339500.0]

Command took 4.48 minutes -- by nahmed27@gnu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cod 35

1 # display the first few rows of the transformed dataset
2 display(transformed_data)

(1) Spark Jobs

Table +

	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALDATE	PRICE	QUALIFIED	GFA	BLDO_NUM	STYLE	...
1	4	0	Warm Cool	Y	2	8	4	1910	1988	1972	3	2003-11-25T00:00:00Z	1095000	Q	2522	1	3 Story	I
2	3	1	Hot Water Rad	Y	2	9	5	1910	2009	1984	3	2016-06-21T00:00:00Z	2100000	Q	2522	1	3 Story	I

AIT 614-SecDL2_Team5_Sys Python v ⚡ skumar2@gmu.edu v

File Edit View Run Help Last edit was 1 hour ago Provide feedback

```
|(146,[0,1,2,3,4,5,...] 139580.0)
|(146,[0,1,2,3,4,5,...] 339580.0)

Command took 4.00 minutes -- by naheed27@gpu.edu at 12/5/2023, 5:45:52 PM on Cluster
```

Cmd 35

```
1 # display the first few rows of the transformed dataset
2 display(transformed_data)
```

× [1] Spark Jobs

Table +

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDO_NUM	STYLE	GR.	
4	0	Warm Cool	Y	2	8	4	1910	1988	1972	3	2003-11-25T00:00:00Z	1095000	Q	2522	1	3 Story	I	
1																		
2	1	Hot Water Rad	Y	2	9	5	1910	2009	1984	3	2016-06-21T00:00:00Z	2100000	Q	2522	1	3 Story	I	
3	1	Hot Water Rad	Y	2	8	5	1900	2003	1984	3	2006-07-12T00:00:00Z	1602000	Q	2484	1	3 Story	I	
4	3	Hot Water Rad	Y	1	10	5	1913	null	1972	4	2010-02-26T00:00:00Z	1950000	Q	5344	1	4 Story	I	
5	3	Hot Water Rad	Y	2	8	4	1906	2011	1972	3	2011-09-29T00:00:00Z	1050000	Q	2401	1	3 Story	I	
6	3	1	Warm Cool	Y	2	7	3	1908	2008	1967	2	2018-05-03T00:00:00Z	1430000	Q	1488	1	2 Story	I

↓ ▾ 1,392 rows | Truncated data | 4.41 seconds runtime

Refreshed 1 hour ago

```
Command took 4.41 seconds -- by naheed27@gpu.edu at 12/5/2023, 5:45:52 PM on Cluster
```

Cmd 36

```
1 # check for NULL Values after feature processing
2 null_counts = transformed_data.select(count(when(col(c).isNull(), c)).alias(c) for c in transformed_data.columns)
3
4 # Display the null counts row
5 display(null_counts)
```

× [2] Spark Jobs

null_counts: pyspark.sql.dataframe.DataFrame = [BATHRM: long, HF_BATHRM: long ... 70 more fields]

Table +

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDO_NUM	STYLE	STRUCT	GR.
1	0	0	0	0	40316	0	0	112	40542	0	40349	1	0	0	40316	0	40316	40316

↓ ▾ 1 row | 16.13 seconds runtime

Refreshed 1 hour ago

```
Command took 16.13 seconds -- by naheed27@gpu.edu at 12/5/2023, 5:45:52 PM on Cluster
```

Cmd 37

```
1 # Create training and test data sets
2 train_df, test_df = transformed_data.randomSplit([0.8, 0.2], seed=42)
3 print(train_df.cache().count())
4 print(test_df.count())
```

× [3] Spark Jobs

train_df: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 70 more fields]
test_df: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 70 more fields]

78797
19419

Command took 32.93 seconds -- by naheed27@gpu.edu at 12/5/2023, 5:45:52 PM on Cluster

Cmd 38

```
1 # Display training data
2 display(train_df)
```

× [2] Spark Jobs

Table +

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDO_NUM	STYLE	STRUCT	GR.
1	0	Hot Water Rad	N	1	0	0	1900	null	1943	2	2015-08-24T00:00:00Z	11000000	U	792	1	2 Story		
2	0	Hot Water Rad	N	1	0	0	1900	null	1943	2	2017-04-11T00:00:00Z	4000000	U	768	1	2 Story		

databricks

AIT 614-SecDL2_Team5_Sys Python w skumara2@gmu.edu

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Command took 32.93 seconds -- by nahed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster Cnd 38

```
1 # Display training data
2 display(train_df)
```

(2) Spark Jobs

Table +

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDO_NUM	STYLE
1	0	Hot Water Rad	N	1	0	0	1900	null	1943	2	2015-08-24T00:00:00Z	11000000	U	792	1	2 Story
2	0	Hot Water Rad	N	1	0	0	1900	null	1943	2	2017-04-11T00:00:00Z	400000	U	768	1	2 Story
3	0	No Data	N	1	0	0	1911	null	1947	2	1999-03-09T00:00:00Z	61538	U	1818	1	2 Story
4	0	No Data	N	1	6	3	1925	null	1943	2	2005-05-05T00:00:00Z	347000	Q	1088	1	2 Story
5	0	No Data	N	2	0	0	1910	null	1943	2	2016-09-12T00:00:00Z	568000	U	1360	1	2 Story
6	1	Air Exchng	N	1	6	3	1900	null	1954	2	1993-04-14T00:00:00Z	45000	Q	1184	1	2 Story

↓ 1,397 rows | Truncated data | 1.49 seconds runtime Refreshed 1 hour ago

Command took 1.49 seconds -- by nahed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster Cnd 39

```
1 # Display test data
2 display(test_df)
```

(2) Spark Jobs

Table +

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDO_NUM	STYLE
0	0	No Data	N	1	0	0	1892	2000	1954	3	2014-01-09T00:00:00Z	2000000	Q	3936	1	3 Story
1	0	Hot Water Rad	N	1	5	2	1900	2012	1978	2	2017-11-09T00:00:00Z	1029000	U	1018	1	2 Story

databricks

AIT 614-SecDL2_Team5_Sys Python w skumara2@gmu.edu

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Command took 1.49 seconds -- by nahed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster Cnd 39

```
1 # Display test data
2 display(test_df)
```

(2) Spark Jobs

Table +

BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	PRICE	QUALIFIED	GBA	BLDO_NUM	STYLE
1	0	No Data	N	1	0	0	1892	2000	1954	3	2014-01-09T00:00:00Z	2000000	Q	3936	1	3 Story
2	0	Hot Water Rad	N	1	5	2	1900	2012	1978	2	2017-11-09T00:00:00Z	1029000	U	1018	1	2 Story
3	1	Elec Base Brd	N	1	2	0	1911	null	1954	1	2013-06-07T00:00:00Z	280000	Q	371	1	1 Story
4	1	Elec Base Brd	N	1	5	2	1900	2002	1964	2	2008-07-08T00:00:00Z	510000	Q	912	1	2 Story
5	1	Evp Cool	Y	1	4	2	1918	1987	1954	2	2010-09-22T00:00:00Z	355000	Q	676	1	2 Story
6	1	Forced Air	N	1	3	1	1900	null	1954	2	1999-06-10T00:00:00Z	140000	U	630	1	2 Story

↓ 1,395 rows | Truncated data | 11.04 seconds runtime Refreshed 1 hour ago

Command took 11.04 seconds -- by nahed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster Cnd 46

Modeling and Prediction

Cmd 41

Random Forest Prediction

Cmd 42

```
1 from pyspark.ml.regression import RandomForestRegressor
2 from pyspark.ml.evaluation import RegressionEvaluator
3
```

Modeling and Prediction -

Random Forest Prediction:

The screenshot shows three stacked Databricks notebooks, each titled "AIT 614-SecDL2_Team5_Sys". The top notebook contains Python code for training a Random Forest Regressor and evaluating it using a RegressionEvaluator. The middle notebook shows the evaluation results, including RMSE, R-squared, and Mean Squared Error. The bottom notebook displays the predicted values for the features, PRICE, and prediction.

```
1 # From pyspark.ml.regression import RandomForestRegressor
2 # From pyspark.ml.evaluation import RegressionEvaluator
3
4 # Create Random Forest object for regression
5 rf_regressor = RandomForestRegressor(featuresCol='features', labelCol='PRICE', numTrees=10, maxDepth=5, seed=42)
6
7 # Fit the Random Forest model to the training data
8 rf_model = rf_regressor.fit(train_df)
9
10 # Make predictions on the test data
11 predictions_rf = rf_model.transform(test_df)
12
13 # Evaluate the model using a regression evaluator - RMSE
14 evaluator = RegressionEvaluator(labelCol='PRICE', predictionCol='prediction', metricName='rmse')
15 rmse = evaluator.evaluate(predictions_rf)
16 print("Root Mean Squared Error:", rmse)
```

```
# Display the RMSE, R-Squared and Mean Squared Error for the Random Forest Regressor Model
# Evaluate the model - RMSE
rf_evaluator_rmse = RegressionEvaluator(labelCol='PRICE', predictionCol='prediction', metricName='rmse')
rf_rmse = rf_evaluator_rmse.evaluate(predictions_rf)
print("Root Mean Squared Error:", rf_rmse)

# Evaluate the model - R-Squared
rf_evaluator_r2 = RegressionEvaluator(labelCol='PRICE', predictionCol='prediction', metricName='r2')
rf_r2 = rf_evaluator_r2.evaluate(predictions_rf)
print("R-squared:", rf_r2)

# Evaluate the model - Mean Squared Error
rf_mse = evaluator.evaluate(predictions_rf, {evaluator.metricName: 'mse'})
print("Mean Squared Error:", rf_mse)
```

```
# Show the predictions for the Random Forest model
predictions_rf.select('features', 'PRICE', 'prediction').show()
```

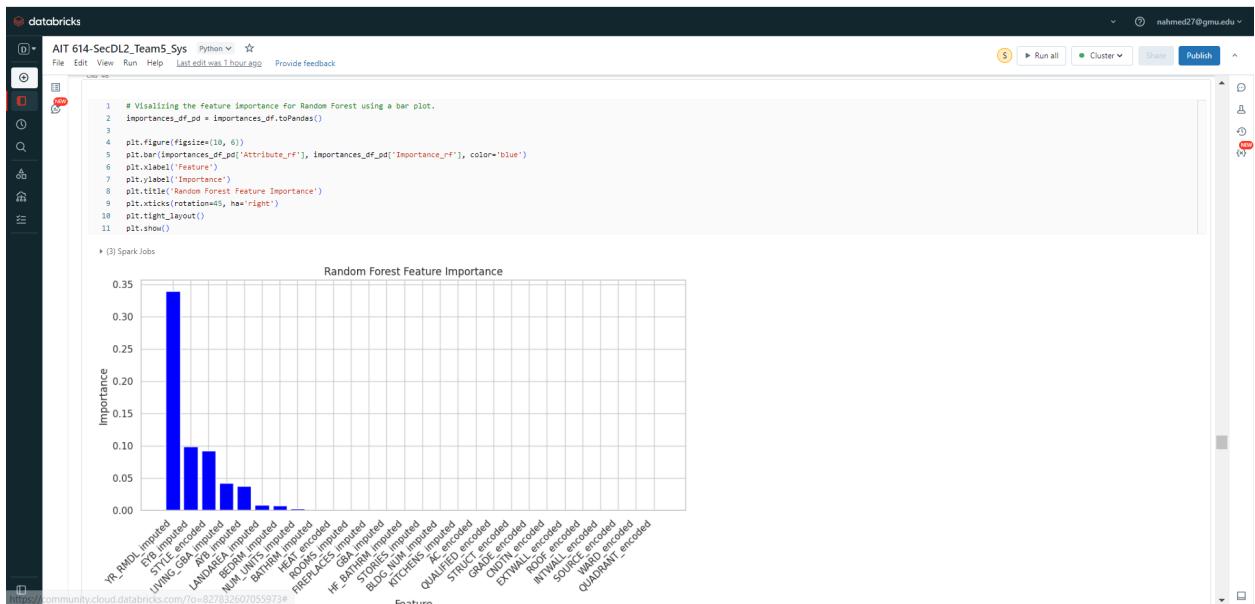
AIT 614-SecDL2_Team5_Sys Python

```

1 #Determine Feature Importance for the Random Forest Model
2 importances_rf = rf_model.featureImportances.toArray()
3
4 # Get feature names from the vector assembler
5 feature_names = assembler_.getInputCol()
6
7 # Create a Pandas DataFrame with feature names and importance scores
8 importances_df_pd_rf = pd.DataFrame(list(zip(feature_names, importances_rf)), columns=["Attribute_rf", "Importance_rf"])
9
10 # Create a Spark DataFrame from the Pandas DataFrame
11 importances_df = spark.createDataFrame(importances_df_pd_rf)
12
13 # Sort the DataFrame by importance in descending order
14 importances_df = importances_df.sort("Importance_rf", ascending=False)
15
16 # Show the result
17 importances_df.show()

+-----+-----+
| Attribute_rf | Importance_rf |
+-----+-----+
| YR_RNDL_imputed | 0.3395122532024931 |
| EYB_imputed | 0.09520000559532025 |
| STYL_imputed | 0.08250000559532025 |
| LIVING_GBA_imputed | 0.0428545888834048 |
| AYB_imputed | 0.03799574935899444 |
| LANDAREA_imputed | 0.008224651899755079 |
| BEDRM_imputed | 0.00769587193515091 |
| NUM_UNITS_imputed | 0.00256300319265... |
| BATHRM_imputed | 0.001642174500000001 |
| HEAT_encoded | 0.5684243932980174 |
| ROOMS_imputed | 4.092164217425467... |
| FIREPLACES_imputed | 4.07857677288024 |
| GBA_imputed | 3.2746708281085051 |
| BE_imputed | 3.2059080645453351 |
| STORES_imputed | 2.8120397290794594... |
| INTWALL_encoded | 0.01 |
| BLDG_NUM_imputed | 0.01 |
| SOURCE_encoded | 0.01 |
+-----+-----+

```



Linear Regression Prediction:

dataricks AIT 614-SecDL2_Team5_Sys Python ▾ Provide feedback

File Edit View Run Help Last edit was 1 hour ago

Linear Regression Prediction

```

1 # Rename the 'PRICE' column to 'label' in both the training and test data sets to run Linear Regression in PySpark
2
3 train_data = train_df.withColumnRenamed("PRICE", "label")
4 test_data = test_df.withColumnRenamed("PRICE", "label")
5
6 # Display training data
7 display(train_data)
8
9 # Display test data
10 display(test_data)

▶ (4) Spark Jobs
▶ (1) predictions: pyspark.sql.DataFrame: [BATHRM: integer, HF_BATHRM: integer ... 70 more fields]
▶ (1) test_data: pyspark.sql.DataFrame: [BATHRM: integer, HF_BATHRM: integer ... 70 more fields]

Table +
```

	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	SALEDATE	label	QUALIFIED	GBA	BLDG_NUM	STYLE
1	0	0	Hot Water Rad	N	1	0	0	1900	null	1943	2	2015-08-24T00:00:00Z	11000000	U	792	1	2 Stor
2	0	0	Hot Water Rad	N	1	0	0	1900	null	1943	2	2017-04-11T00:00:00Z	400000	U	768	1	2 Stor
3	0	0	No Data	N	1	0	0	1911	null	1947	2	1999-03-09T00:00:00Z	61538	U	1818	1	2 Stor
4	0	0	No Data	N	1	6	3	1925	null	1943	2	2005-05-05T00:00:00Z	347000	Q	1068	1	2 Stor
5	0	0	No Data	N	2	0	0	1910	null	1942	2	2016-09-12T00:00:00Z	568000	U	1360	1	2 Stor
6	1	0	Air Exchng	N	1	6	3	1900	null	1954	2	1993-04-14T00:00:00Z	45000	Q	1184	1	2 Stor
7	1	0	Elec Base Brd	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
8	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
9	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
10	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
11	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
12	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
13	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
14	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
15	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
16	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
17	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
18	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
19	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
20	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
21	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
22	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
23	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
24	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
25	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
26	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
27	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
28	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
29	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
30	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
31	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
32	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
33	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
34	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
35	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
36	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
37	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
38	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
39	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
40	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
41	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
42	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
43	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
44	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
45	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
46	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
47	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
48	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
49	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
50	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
51	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
52	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
53	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
54	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
55	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
56	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
57	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
58	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
59	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
60	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
61	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
62	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
63	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
64	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
65	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
66	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
67	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
68	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
69	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
70	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
71	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
72	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
73	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
74	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
75	1	0	No Data	N	1	3	1	1890	null	1954	2	2000-05-15T00:00:00Z	125000	U	528	1	2 Stor
76	1	0	No Data	N	1	3	1	1890	null	1954	2						

AIT 614-SecDL2_Team5_Sys Python

```

1 # Display the RMSE, R-Squared and Mean Squared Error for the Linear Regression Model
2
3 # Evaluate the model - RMSE
4 reg_evaluator_rmse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
5 regression_rmse = reg_evaluator_rmse.evaluate(predictions)
6
7 print(f"Root Mean Squared Error (RMSE): {regression_rmse}")
8
9 # Evaluate the model - R-Squared
10 reg_evaluator_r2 = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="r2")
11 regression_r2 = reg_evaluator_r2.evaluate(predictions)
12
13 print(f"R-squared: {regression_r2}")
14
15 # Evaluates the model - Mean Squared Error
16 reg_evaluator_mse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="mse")
17 regression_mse = reg_evaluator_mse.evaluate(predictions)
18 print(f"Mean Squared Error (MSE): {regression_mse}")

▶ (6) Spark Jobs
Root Mean Squared Error (RMSE): 6299361.55697987
R-squared: 0.0346762659807474
Mean Squared Error (MSE): 39681956025442.47
Command took 1.78 minutes -- by nahmed27@gnu.edu at 12/5/2023, 5:45:52 PM on Cluster

```

AIT 614-SecDL2_Team5_Sys Python

```

1 # Show the predictions for the Linear Regression model
2 predictions.select("features", "label", "prediction").show()

▶ (1) Spark Jobs
+-----+-----+-----+
|features|label|prediction|
+-----+-----+-----+
|(446,[2,3,4,5,6,...,12000000,0])|6463599.8534480015| 6463599.8534480015| |
|(446,[1,2,3,4,5,6,...,1829300,0])|2089795.2586361766| 2089795.2586361766|
|(446,[0,2,3,4,5,6,...,7,...])|280000,0|38824.0259819323| 38824.0259819323|
|(446,[0,2,3,4,5,6,...,1])|5180000,0|924.7440356973078| 924.7440356973078|
|(446,[0,2,3,4,5,6,...,1])|3558000,0|-1213931.419428274| -1213931.419428274|
|(446,[0,2,3,4,5,6,...,1])|2089795.2586361766|210551.8012819721| 210551.8012819721|
|(446,[0,2,3,4,5,6,...,1])|2089795.2586361766|210551.8012819721| 210551.8012819721|
|(446,[0,2,3,4,5,6,...,1])|4858000,0|-454772.1511932777| -454772.1511932777|
|(446,[0,2,3,4,5,6,...,1])|2058000,0|1122720.306994346| 1122720.306994346|
|(446,[0,2,3,4,5,6,...,1])|5580000,0|1056758.3949712962| 1056758.3949712962|
|(446,[0,2,3,4,5,6,...,1])|808000,0|-1083799.3132812321| -1083799.3132812321|
|(446,[0,2,3,4,5,6,...,1])|356000,0|-394280.6316551267| -394280.6316551267|
|(446,[0,2,3,4,5,6,...,1])|1260000,0|-511471.3338773941| -511471.3338773941|
|(446,[0,2,3,4,5,6,...,1])|4690000,0|-493237.366238458| -493237.366238458|
|(446,[0,2,3,4,5,6,...,1])|758000,0|1553932.7933481464| 1553932.7933481464|
|(446,[0,1,2,3,4,5,6,...,1])|2280000,0|1796865.1398637377| 1796865.1398637377|
|(446,[0,2,3,4,5,6,...,1])|838000,0|734523.5661774807| 734523.5661774807|
Command took 11.29 seconds -- by nahmed27@gnu.edu at 11/5/2023, 5:45:52 PM on Cluster

```

AIT 614-SecDL2_Team5_Sys Python

```

1 #Determine the Feature importance For the Linear Regression Model (pyspark)
2
3 # Get coefficients from the model
4 coefficients = lr_model.coefficients.toArray()
5
6 # Get feature names from the vector assembler
7 feature_name_lr = assembler.getOutputCols()
8
9 # Create a Pandas DataFrame with feature name and coefficients
10 importances_df_pd_lr = pd.DataFrame(list(zip(feature_name_lr, coefficients)), columns=["Attribute_lr", "Coefficient_lr"])
11
12 # Sort the DataFrame by absolute coefficient values in descending order
13 importances_of_pd_lr = importances_df_pd_lr.assign(Absolute_Coefficient_lr=lambda x: abs(x['Coefficient_lr']))
14 importances_of_pd_lr = importances_of_pd_lr.sort_values(by='Absolute_Coefficient_lr', ascending=False)
15
16 print(importances_of_pd_lr)

Attribute_lr Coefficient_lr Absolute_Coefficient_lr
15 STRUCT_encoded -1.230000e+05 1.230000e+05
17 QUALIF_LR_encoded -4.300054e+05 4.300054e+05
18 BLDR_BATHR_encoded 9.724192e+05 9.724192e+05
20 GRADE_encoded -7.163732e+05 7.163732e+05
27 QUADRANT_encoded 7.238429e+05 7.238429e+05
23 ROOF_encoded -6.115887e+05 6.115887e+05
25 SOURCE_encoded 5.396937e+05 5.396937e+05
19 HUM_UNIT_Ln_imputed 3.120000e+05 3.120000e+05
17 QUALIF_LL_encoded 3.176772e+05 3.176772e+05
21 CHTH_AC_encoded -3.178746e+05 3.178746e+05
16 CHDV_AC_encoded -3.479394e+05 3.479394e+05
15 HEAT_encoded 3.276421e+05 3.276421e+05
26 HARD_encoded 3.027639e+05 3.027639e+05
24 ELEC_AC_imputed 2.800000e+05 2.800000e+05
2 HUM_UNIT_LL_imputed 3.176772e+05 3.176772e+05
22 EXTHLL_AC_encoded -1.556325e+05 1.556325e+05
1 HF_BATHR_imputed 1.244792e+05 1.244792e+05
3 ROOMS_imputed -8.407390e+04 8.407390e+04
4 BEDR_imputed -7.921775e+04 7.921775e+04
11 KITCHNS_imputed 7.880150e+04 7.880150e+04
6 YR_RIVL_imputed 2.072083e+04 2.072083e+04
Command took 8.15 seconds -- by nahmed27@gnu.edu at 12/5/2023, 5:45:12 PM on Cluster

```

AIT 614-SecDL2_Team5_Sys Python ▾ ☆

Last edit was 1 hour ago Provide feedback

File Edit View Run Help

Run all Cluster Share Publish

Cmd 33

```

1 # Visualizing the Feature importance For Linear Regression using a bar plot.
2 importances_df_pd_lr = importances_df_pd_lr.sort_values(by='Absolute_Coefficient_lr', ascending=False)
3
4 plt.figure(figsize=(10, 6))
5 plt.bar(importances_df_pd_lr['Attribute_lr'], importances_df_pd_lr['Absolute_Coefficient_lr'], color='green')
6 plt.xlabel('Feature')
7 plt.ylabel('Absolute Coefficient')
8 plt.title('Linear Regression Feature Importance')
9 plt.xticks(rotation=45, ha='right')
10 plt.tight_layout()
11 plt.show()

```

Linear Regression Feature Importance

Feature	Absolute Coefficient
STILE encoded	~1.5e6
BLDG_NUM_imputed	~0.9e6
QUAIRAB encoded	~0.75e6
ROOF encoded	~0.7e6
SOURCE encoded	~0.6e6
QUALIFIED encoded	~0.55e6
CFDTH encoded	~0.5e6
HCR encoded	~0.45e6
WARD encoded	~0.4e6
NTWALL encoded	~0.35e6
EXWALL encoded	~0.3e6
HE encoded	~0.25e6
BATHRM_imputed	~0.2e6
BEDRM_imputed	~0.15e6
KITCHNS_imputed	~0.1e6
LR_RMDL_imputed	~0.05e6
EVB_imputed	~0.02e6
BATHRM_imputed	~0.01e6
STORES_imputed	~0.005e6
FIREPLACES_imputed	~0.002e6
GBA_imputed	~0.001e6
LANDAREA_imputed	~0.0005e6

Cmd 34

```

1 #Here is a simple linear regression model using sklearn that prints the accuracy and r^2 score in comparison to the other one. It looks like that the linear regression algorithm is not a good fit to this dataset.
2
3 from sklearn.model_selection import train_test_split
4 from sklearn.linear_model import LinearRegression
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 # 'PRICE' is the target variable
8 #Selected few categorical features based on the dataset
9 selected_features = ['BATHRM', 'HF_BATHRM', 'ROOMS', 'BEDRM', 'YR_RHDL', 'EVB', 'STORIES', 'GBA', 'FIREPLACES', 'LANDAREA']
10
11 # Filtering the PRICE column
12 data_for_modeling = housing_data_pd[selected_features + ['PRICE']].copy()
13
14 # Dropped the rows with missing values
15 data_for_modeling.dropna(inplace=True)
16
17 # Split the data into training and testing sets (80%:20%)
18 X = data_for_modeling[selected_features]
19 y = data_for_modeling['PRICE']
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
21
22 # fitting the training set into the Linear Regression model
23 model = LinearRegression()
24 model.fit(X_train, y_train)
25
26 # Predictions
27 y_pred = model.predict(X_test)
28
29
30 #Accuracy
31 accuracy = model.score(X_test, y_test)
32 print("Model Accuracy: ({accuracy})")
33
34 # Model evaluation
35 mse = mean_squared_error(y_test, y_pred)
36 r2 = r2_score(y_test, y_pred)
37
38
39 print("Mean Squared Error (MSE): (mse)")
40 print("R-squared (R2): (r2)")

```

File Edit View Run Help Last edit was 1 hour ago Provide feedback

```

8 #Selected few categorical features based on the dataset
9 selected_features = ['BATHRM', 'HF_BATHRM', 'ROOMS', 'BEDRM', 'AVB', 'VR_BDRL', 'EVB', 'STORIES', 'GFA', 'FIREPLACES', 'LANDAREA']
10
11 # Filtering the PRICE column
12 data_for_modeling = housing_data_pd[selected_features + ['PRICE']].copy()
13
14 # Dropped the rows with missing values
15 data_for_modeling.dropna(inplace=True)
16
17 # Split the data into training and testing sets (80%/20%)
18 X = data_for_modeling[selected_features]
19 y = data_for_modeling[['PRICE']]
20 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
21
22 # Fitting the training set into the Linear Regression model
23 model = LinearRegression()
24 model.fit(X_train, y_train)
25
26 # Predictions
27 y_pred = model.predict(X_test)
28
29
30 #accuracy
31 accuracy = model.score(X_test, y_test)
32 print("Model Accuracy: " + str(accuracy))
33
34 # Model evaluation
35 mse = mean_squared_error(y_test, y_pred)
36 r2 = r2_score(y_test, y_pred)
37
38
39 print("Mean Squared Error (MSE): " + str(mse))
40 print("R-squared (R2): " + str(r2))

+ (1) MLflow run
    Logged 1 run to an experiment in MLflow. Learn more
Model Accuracy: 0.554579682733035
Mean Squared Error (MSE): 1681084236088.51816
R-squared (R2): 0.554579682733035
Command took 6.09 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

```

File Edit View Run Help Last edit was 1 hour ago Provide feedback

```

Command took 6.09 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster
Cnd 35

1 # Predictions for the model
2 display(y_pred)

array([920806.48304782, 754705.06306251, 592585.06539384, ...,
       338331.9382072, 156895.57523591, 475153.680287283])
Command took 0.09 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

```

Decision Tree Prediction:

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Decision Tree Prediction

```

Cnd 37

1 from pyspark.ml.regression import DecisionTreeRegressor
2 from pyspark.ml.evaluation import RegressionEvaluator
3
4 # Create a Decision Tree Regressor
5 dt = DecisionTreeRegressor(featuresCol="features", labelCol="label")
6
7 # Fit the model to the training data
8 dt_model = dt.fit(train_data)
9
10 # Make predictions on the test data
11 predictions = dt_model.transform(test_data)
12
13 # Evaluate RMSE
14 reg_evaluator_rmse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
15 regression_rmse = reg_evaluator_rmse.evaluate(predictions)
16 print("Root Mean Squared Error (RMSE): " + str(regression_rmse))
17
18 # Evaluate R-Squared
19 reg_evaluator_r2 = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName='r2')
20 regression_r2 = reg_evaluator_r2.evaluate(predictions)
21 print("R-squared: " + str(regression_r2))
22
23 # Evaluate Mean Squared Error
24 reg_evaluator_mse = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="mse")
25 regression_mse = reg_evaluator_mse.evaluate(predictions)
26 print("Mean Squared Error (MSE): " + str(regression_mse))

+ (19) Spark Jobs
+ (1) MLflow run
    Logged 1 run to an experiment in MLflow. Learn more
+ (1) predictions: pyspark.sql.dataframe.DataFrame = [BATHRM: integer, HF_BATHRM: integer ... 71 more fields]
Root Mean Squared Error (RMSE): 2283841.388734778
R-squared: 0.8734387344827492
Mean Squared Error (MSE): 521227982616.823
Command took 4.08 minutes -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster

```

```

Cnd 58

1 # Show the predictions for the Decision Tree model
2 predictions.select('label', 'prediction').show()

+ (1) Spark Jobs

```

dataricks

AIT 614-SecDL2_Team5_Sys Python ⚡ Provide feedback

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Logged In to an experiment in MLflow. Learn more

→ predictions: pySpark DataFrame = [BATHRM: Integer, HF_BATHRM: Integer ... 71 more fields]

R-mean: Mean Squared Error (MSE): 2283841.388734778

R-squared: 0.8723873448027493

Mean Squared Error (MSE): 521227982676.023

Command took 4:07 minutes — by named27@gnu.edu at 12/5/2023, 5:45:52 PM on Cluster

End 58

Python ► ▾ x

```
1 # Show the predictions for the Decision Tree model
2 predictions.select("features", "label", "prediction").show()
3
4 [1] Spark Jobs
5
6 +-----+-----+-----+
7 | features| label| prediction|
8 +-----+-----+-----+
9 |([4, 5, 6, 8, 9, -1], 2000000, 0) | 3698991.9982344666 |
10 |([1, 2, 3, 4, 5, 6, -1], 1829800, 0) | 3143341.283636337 |
11 |([4, 6, 2, 3, 5, 6, -1], 288000, 0) | 285919.7821552723 |
12 |([6, 2, 3, 4, 5, 6, -1], 518000, 0) | 439424.38581708361 |
13 |([4, 6, 2, 3, 4, 5, 6, -1], 128000, 0) | 3698991.9982344666 |
14 |([6, 2, 3, 4, 5, 6, -1], 148000, 0) | 3698991.9982344666 |
15 |([6, 2, 3, 4, 5, 6, -1], 208000, 0) | 285919.7821552723 |
16 |([6, 2, 3, 4, 5, 6, -1], 485000, 0) | 439424.38581708361 |
17 |([6, 2, 3, 4, 5, 6, -1], 205000, 0) | 285919.7821552723 |
18 |([6, 2, 3, 4, 5, 6, -1], 137000, 0) | 3698991.9982344666 |
19 |([6, 2, 3, 4, 5, 6, -1], 88000, 0) | 439424.38581708361 |
20 |([6, 2, 3, 4, 5, 6, -1], 137000, 0) | 3698991.9982344666 |
21 |([6, 2, 3, 4, 5, 6, -1], 358000, 0) | 285919.7821552723 |
22 |([6, 2, 3, 4, 5, 6, -1], 128000, 0) | 3698991.9982344666 |
23 |([6, 2, 3, 4, 5, 6, -1], 128000, 0) | 285919.7821552723 |
24 |([6, 2, 3, 4, 5, 6, -1], 75000, 0) | 285919.7821552723 |
25 |([6, 2, 3, 4, 5, 6, -1], 228000, 0) | 285919.7821552723 |
26 |([6, 2, 3, 4, 5, 6, -1], 83000, 0) | 439424.38581708361 |
27
28 Command took 21.93 seconds — by named27@gnu.edu at 12/5/2023, 5:45:52 PM on Cluster
```

End 59

Python ► ▾ x

```
1 # Get feature importances from the model
2 feature_importances = dt_model.featureImportances.toArray()
3
4 # Get feature names from the vector assembler
5 feature_names_dt = assembler.getInputCols()
6
7 # Create a Pandas DataFrame with feature names and importances
8 importances_dt_pd_dt = pd.DataFrame(list(zip(feature_names_dt, feature_importances)), columns=["Attribute_dt", "Importance_dt"])
9
10 # Sort the DataFrame by importance values in descending order
11 importances_dt_pd_dt = importances_dt_pd_dt.sort_values(by="Importance_dt", ascending=False)
12
13 print(importances_dt_pd_dt)
```

dataricks

AIT 614-SecDL2_Team5_Sys Python ⚡ Provide feedback

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Logged In to an experiment in MLflow. Learn more

→ predictions: pySpark DataFrame = [BATHRM: Integer, HF_BATHRM: Integer ... 71 more fields]

R-mean: Mean Squared Error (MSE): 2283841.388734778

R-squared: 0.8723873448027493

Mean Squared Error (MSE): 521227982676.023

Command took 4:07 minutes — by named27@gnu.edu at 12/5/2023, 5:45:52 PM on Cluster

End 59

Python ► ▾ x

```
1 # Get feature importances from the model
2 feature_importances = dt_model.featureImportances.toArray()
3
4 # Get feature names from the vector assembler
5 feature_names_dt = assembler.getInputCols()
6
7 # Create a Pandas DataFrame with feature names and importances
8 importances_dt_pd_dt = pd.DataFrame(list(zip(feature_names_dt, feature_importances)), columns=["Attribute_dt", "Importance_dt"])
9
10 # Sort the DataFrame by importance values in descending order
11 importances_dt_pd_dt = importances_dt_pd_dt.sort_values(by="Importance_dt", ascending=False)
12
13 print(importances_dt_pd_dt)
```

AIT 614-SecDL2_Team5_Sys Python ⚡ Provide feedback

File Edit View Run Help Last edit was 1 hour ago Provide feedback

Logged In to an experiment in MLflow. Learn more

→ predictions: pySpark DataFrame = [BATHRM: Integer, HF_BATHRM: Integer ... 71 more fields]

R-mean: Mean Squared Error (MSE): 2283841.388734778

R-squared: 0.8723873448027493

Mean Squared Error (MSE): 521227982676.023

Command took 4:07 minutes — by named27@gnu.edu at 12/5/2023, 5:45:52 PM on Cluster

End 60

Python ► ▾ x

```
1 # Get feature importances from the model
2 feature_importances = dt_model.featureImportances.toArray()
3
4 # Get feature name from the vector assembler
5 feature_names_dt = assembler.getInputCols()
6
7 # Create a Pandas DataFrame with feature names and importances
8 importances_dt_dt_dt = pd.DataFrame(list(zip(feature_names_dt, feature_importances)), columns=["Attribute", "Importance"])
```

skumara2@gmu.edu

```

AIT 614-SecDL2_Team5_Sys Python v ☆ skumara2@gmu.edu
File Edit View Run Help Last edit was 1 hour ago Provide feedback
Run all Cluster Share Publish
1 # Get feature importances from the model
2 feature_importances = dt_model.featureImportances.toArray()
3
4 # Get feature names from the vector assembler
5 feature_names_dt = assembler.getInputCols()
6
7 # Create a Pandas DataFrame with feature names and importances
8 importances_df_pd_dt = pd.DataFrame(list(zip(feature_names_dt, feature_importances)), columns=["Attribute", "Importance"])
9
10 # Sort the DataFrame by importance values in descending order
11 top_importances_df_pd_dt = importances_df_pd_dt.sort_values(by="Importance", ascending=False).head(10)
12
13 # Plotting
14 plt.figure(figsize=(10, 6))
15 plt.bar(top_importances_df_pd_dt['Attribute'], top_importances_df_pd_dt['Importance'], color='blue')
16 plt.xlabel('Features')
17 plt.ylabel('Importance')
18 plt.title('Feature Importances from Decision Tree')
19 plt.xticks(rotation=45)
20 plt.show()

Feature Importances from Decision Tree

```

Feature	Importance
imputed	~0.38
imputed	~0.11
encoded	~0.04
imputed	~0.03
imputed	~0.02
imputed	~0.01

skumara2@gmu.edu

```

AIT 614-SecDL2_Team5_Sys Python v ☆ skumara2@gmu.edu
File Edit View Run Help Last edit was 1 hour ago Provide feedback
Run all Cluster Share Publish
U.UU
Features
Command took 0.68 seconds -- by nahmed27@gmu.edu at 12/5/2023, 5:45:52 PM on Cluster
Cmd 61
1 # Get the decision tree model
2 tree_model = dt_model
3
4 # Print out the decision tree
5 print(tree_model.toDebugString)

DecisionTreeRegressionModel: uid=DecisionTreeRegressor_f93ff58a01a1, depth=5, numNodes=57, numFeatures=146
If (feature 136 in {0,0})
  If (feature 135 in {0,0})
    If (feature 72 in {0,0})
      If (feature 0 <= 2.5)
        If (feature 41 in {1,0})
          Predict: 285919.7821552723
        Else (feature 74 not in {1,0})
          Predict: 439424.30581708136
        Else (feature 0 > 2.5)
          If (feature 7 > 1968.5)
            Predict: 522396.62291853176
          Else (feature 7 > 1968.5)
            Predict: 763864.6055111111
          Else (feature 72 not in {0,0})
            Predict: 2.587
      Else (feature 135 not in {0,0})
        If (feature 6 < 2085.5)
          If (feature 0 <= 2.5)
            If (feature 41 in {1,0})
              Predict: 369889.9982844666

```

Shift+Enter to run
Shift+Ctrl+Enter to run selected text