Where to Begin

FEATURE ENGINEERING WITH PYSPARK

John Hogue
Lead Data Scientist, General Mills

Diving Straight to Analysis

Here be Monsters

- Become your own expert
- Define goals of analysis
- Research your data
- Be curious, ask questions

The Data Science Process

Spark changes fast and frequently

- Latest documentation:
  - https://spark.apache.org/docs/latest/
- Specific version (2.3.1)
  - https://spark.apache.org/docs/2.3.1/
- Check your versions!

```
# return spark version
spark.version

# return python version
import sys
sys.version_info
```

Data Formats: Parquet

Data is supplied as Parquet

- Stored Column-wise
  - Fast to query column subsets
- Structured, defined schema
  - Fields and Data Types defined
  - Great for messy text data
- Industry Adopted
  - Good skill to have! ????

Getting the Data to Spark

PySpark `read` methods

- PySpark supports many file types!

```
# JSON
spark.read.json('example.json')
```

```
# CSV or delimited files
spark.read.csv('example.csv')
```

```
# Parquet
spark.read.parquet('example.parq')
```

```
# Read a parquet file to a PySpark DataFrame
df = spark.read.parquet('example.parq')
```

Let's Practice!

FEATURE ENGINEERING WITH PYSPARK

Defining A Problem

FEATURE ENGINEERING WITH PYSPARK

John Hogue
Lead Data Scientist, General Mills

What's Your Problem?

Predict the selling price of a house

- Given is listed price and features
  - X, independent 'known' variables
- How much to buy the house for
  - Y, dependent 'unknown' variable
  - `SALESCLOSEPRICE`

Context & Limitations of our Real Estate

- **Homes sold St Paul, MN Area**
  - Includes several suburbs
- **Real Estate Types**
  - Residential-Single
  - Residential-Multi-Family
- **Full Year of Data**
  - Impact of seasonality

What types of attributes are available?

- **Dates**
  - Date Listed
  - Year Built
- **Location**
  - City
  - School District
  - Address
- **Size**
  - # Bedrooms & Bathrooms
  - Living Area
- **Price**
  - List Price
  - Sales Closing Price
- **Amenities**
  - Pool
  - Fireplace
  - Garage
- **Construction Materials**
  - Siding
  - Roofing

Validating Your Data Load

- `DataFrame.count()` for row count

```
df.count()
```

```
5000
```

  - `DataFrame.columns` for a list of columns

```
df.columns
```

```
['No.', 'MLSID', 'StreetNumberNumeric', ... ]
```

  - Length of `DataFrame.columns` for the number of columns

```
len(df.columns)
```

```
74
```

Checking Datatypes

`DataFrame.dtypes`

- Creates a list of columns and their data types tuples

```
df.dtypes
```

```
[('No.', 'integer'), ('MLSID', 'string'), ... ]
```

Let's Practice

FEATURE ENGINEERING WITH PYSPARK

Visually Inspecting Data

FEATURE ENGINEERING WITH PYSPARK

John Hogue
Lead Data Scientist, General Mills

Getting Descriptive with DataFrame.describe()

```
df.describe(['LISTPRICE']).show()
```

```
+-------+------------------+
|summary|         LISTPRICE|
+-------+------------------+
|  count|              5000|
|   mean|         263419.365|
| stddev|143944.10818036905|
|    min|            100000|
|    max|             99999|
+-------+------------------+
```

Many descriptive functions are already available

- Mean
  - `pyspark.sql.functions.mean(col)`
- Skewness
  - `pyspark.sql.functions.skewness(col)`
- Minimum
  - `pyspark.sql.functions.min(col)`
- Covariance
  - `cov(col1, col2)`
- Correlation
  - `corr(col1, col2)`

Example with mean()

- `mean(col)`
- Aggregate function: returns the average (mean) of the values in a group.

```
df.agg({'SALESCLOSEPRICE': 'mean'}).collect()
```

```
[Row(avg(SALESCLOSEPRICE)=262804.4668)]
```

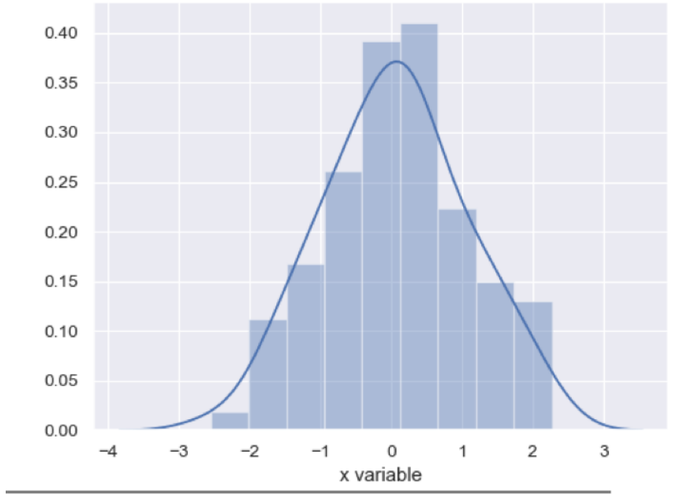
Example with cov()

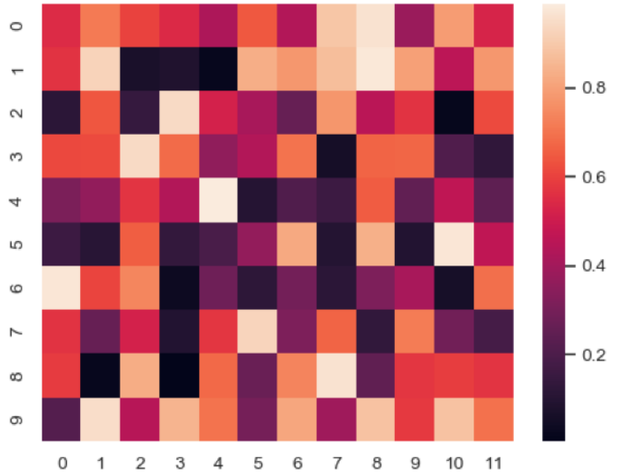
- `cov(col1, col2)`
- Parameters:
  - col1 – first column
  - col2 – second column

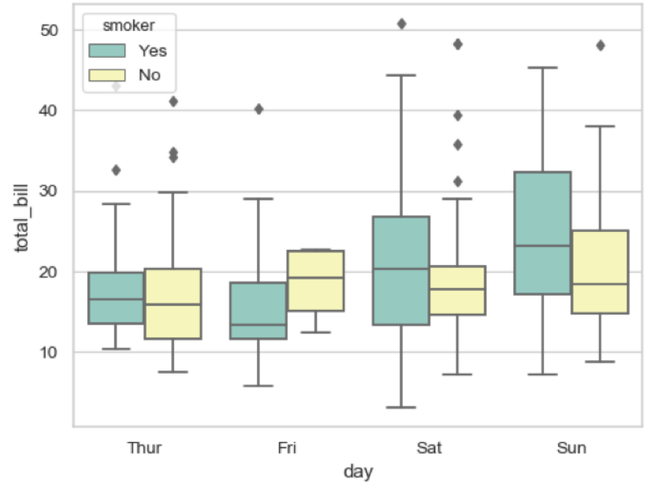
```
df.cov('SALESCLOSEPRICE', 'YEARBUILT')
```

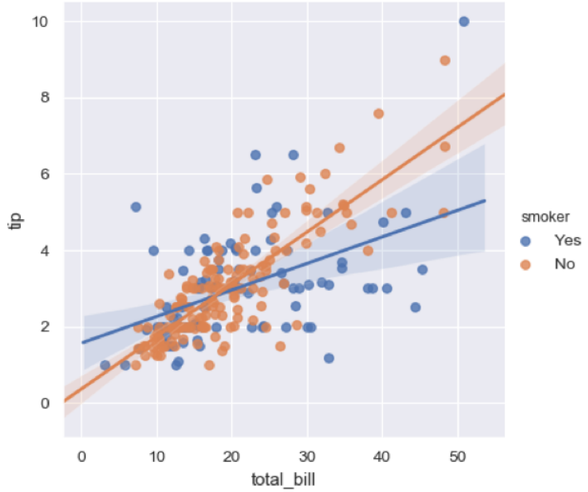
```
1281910.3840634783
```

seaborn: statistical data visualization









Notes on plotting

Plotting PySpark DataFrames using standard libraries like Seaborn require conversion to Pandas

WARNING: *Sample PySpark DataFrames before converting to Pandas!*

- `sample(withReplacement, fraction, seed=None)`
  - `withReplacement` allow repeats in sample
  - `fraction` % of records to keep
  - `seed` random seed for reproducibility

```
# Sample 50% of the PySpark DataFrame and count rows
df.sample(False, 0.5, 42).count()
```

```
2504
```

Prepping for plotting a distribution

Seaborn `distplot()`

- `seaborn.distplot(a)`
- `a` : Series, 1d-array, or list. Observed data.

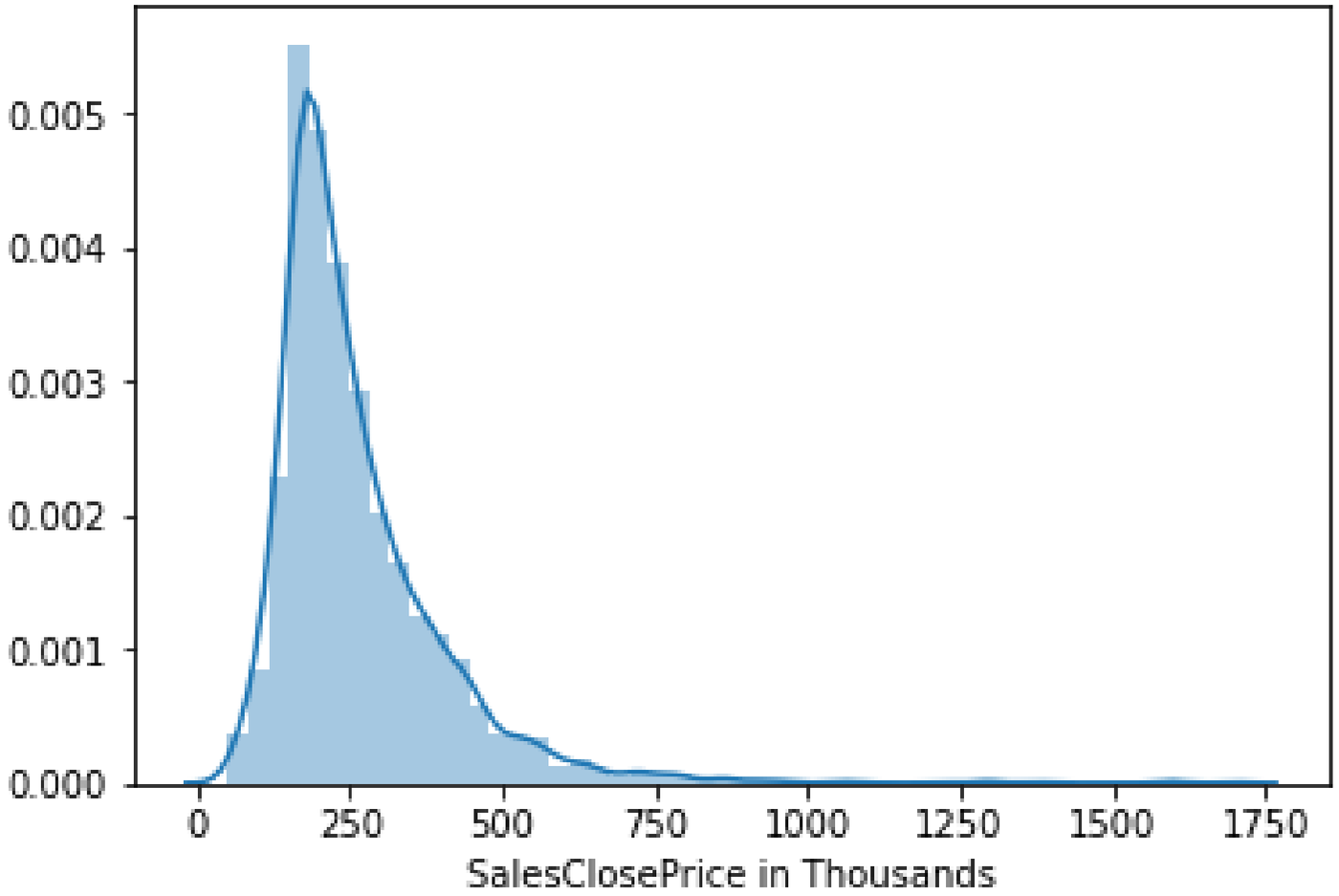
```
# Import your favorite visualization library
import seaborn as sns

# Sample the dataframe
sample_df = df.select(['SALESCLOSEPRICE']).sample(False, 0.5, 42)

# Convert the sample to a Pandas DataFrame
pandas_df = sample_df.toPandas()

# Plot it
sns.distplot(pandas_df)
```

Distribution plot of sales closing price



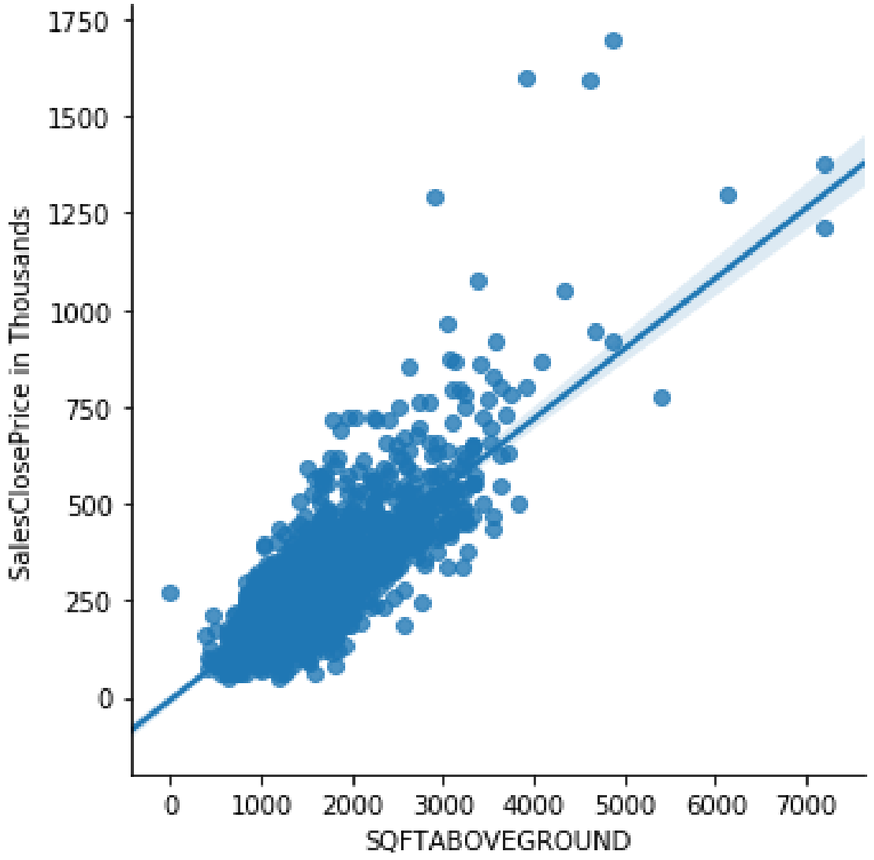
Relationship plotting

Seaborn `lmplot()`

- `seaborn.lmplot(x, y, data)`
- `x`, `y` : strings, Input variables; these should be column names in data.
- `data` : Pandas DataFrame

```
# Import your favorite visualization library
import seaborn as sns
# Select columns
s_df = df.select(['SALESCLOSEPRICE', 'SQFTABOVEGROUND'])
# Sample dataframe
s_df = s_df.sample(False, 0.5, 42)
# Convert to Pandas DataFrame
pandas_df = s_df.toPandas()
# Plot it
sns.lmplot(x='SQFTABOVEGROUND', y='SALESCLOSEPRICE', data=pandas_df)
```

Linear model plot between SQFT above ground and sales price



Let's practice!

FEATURE ENGINEERING WITH PYSPARK