

ADVANCED BIG DATA

SAN FRANCISCO CRIME CLASSIFICATION Using SparkML

A Project Report

Submitted by

BHOOMIKA S BABU - PES1PG21CS010

CHAITHANYA V - PES1PG21CS011

1. STREAMING DATA:

Streaming data is data that is generated continuously by thousands of data sources, which is typically send in the data records simultaneously, and in small sizes.

Converting from RDD to DataFrames:

```
C:\Windows\System32\cmd.exe - python3 stream.py -f train.csv -b 1000
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL\Desktop>python3 stream.py -f train.csv -b 1000
Namespace(file='train.csv', batch_size=1000, endless=False)
Waiting for connection on port 6100...
Connected to ('127.0.0.1', 60718)

| 44/878 [00:44:14:03, 1.01s/it]
```

```
C:\Windows\System32\cmd.exe - python3 crimeReport1.py
22/08/27 01:51:16 WARN BlockManager: Block input-0-1661545275800 replicated to only 0 peer(s) instead of 1 peers
22/08/27 01:51:17 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
22/08/27 01:51:17 WARN BlockManager: Block input-0-1661545276800 replicated to only 0 peer(s) instead of 1 peers
22/08/27 01:51:18 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
22/08/27 01:51:18 WARN BlockManager: Block input-0-1661545277800 replicated to only 0 peer(s) instead of 1 peers

+-----+-----+-----+-----+-----+-----+-----+
| Date | Category | Description | dayOfweek | District | Resolution | address |
+-----+-----+-----+-----+-----+-----+-----+
| [2015-05-01 17:09:00] | NON-CRIMINAL | AIDED CASE | Friday | SOUTHERN | NONE | 5TH ST / HARRISON ST | -122.401846367522|37.7790324136251|
| [2015-05-01 17:00:00] | OTHER OFFENSES | MISCELLANEOUS INV... | Friday | PARK | NONE | 1700 Block of HAL... | -122.45131444779699|37.769504666162|
| [2015-05-01 17:00:00] | NON-CRIMINAL | FOUND PROPERTY | Friday | PARK | NONE | 1700 Block of HAL... | -122.45131444779699|37.769504666162|
| [2015-05-01 17:00:00] | LARCENY/THEFT | GRAND THEFT FROM ... | Friday | RICHMOND | NONE | 700 Block of 7TH AV | -122.46494471615601|37.774384171507|
| [2015-05-01 17:00:00] | LARCENY/THEFT | PETTY THEFT SHOPL... | Friday | SOUTHERN | NONE | 800 Block of MARK... | -122.40652898714399|37.7850629421661|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
22/08/27 01:51:19 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
22/08/27 01:51:19 WARN BlockManager: Block input-0-1661545278800 replicated to only 0 peer(s) instead of 1 peers
22/08/27 01:51:20 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
22/08/27 01:51:20 WARN BlockManager: Block input-0-1661545279800 replicated to only 0 peer(s) instead of 1 peers
```


The screenshot shows a Google Colab notebook titled 'Untitled24.ipynb'. The left sidebar displays a file explorer with 'sample_data' and 'train.csv'. The main code area contains three cells:

```
[1] pip install pyspark
```

looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>
collecting pyspark
Downloading pyspark-3.3.0.tar.gz (281.3 MB)
[REDACTED] 281.3 MB 45 KB/s
Collecting py4j==0.10.9.5
Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 KB)
[REDACTED] 199 KB 49.0 MB/s
Building wheels for collected packages: pyspark
Building wheel for pyspark (setup.py) ... done
Created wheel for pyspark: filename=pyspark-3.3.0-py2.py3-none-any.whl size=281764026 sha256=b3bcc166253443dc84418d1a1a962e6fb831d5eaca2d33a9b7
Stored in directory: /root/.cache/pip/wheels/7a/8e/1b/f73a52650d2e5f337708d9f6a1750d451a7349a867f928b885
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.0

```
[2] from pyspark.sql import SQLContext
from pyspark import SparkContext
from pyspark.sql import SparkSession
sc = SparkContext()
sqlContext = SQLContext(sc)
spark = SparkSession.builder.appName("BCrimeClass").getOrCreate()
data = spark.read.csv('/content/train.csv', header=True)

/usr/local/lib/python3.7/dist-packages/pyspark/sql/context.py:114: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() inst
FutureWarning,
```

```
[3] drop_list = ['Category', 'Descript', 'Resolution']
```

The bottom of the notebook shows the Windows taskbar with the date 27-08-2022 and time 01:15.

Step-3: Dropping the attributes

The screenshot shows the same Google Colab notebook at a later stage. The code cells are:

```
[3] drop_list = ['Category', 'Descript', 'Resolution']
data = data.select([column for column in data.columns if column not in drop_list])
data.show(5)
```

	Dates	DayOfWeek	PdDistrict	Address	X	Y
[2015-05-13 23:53:00]	Wednesday	NORTHERN	OAK ST / LAGUNA ST	-122.425891675136	37.7745985956747	
[2015-05-13 23:53:00]	Wednesday	NORTHERN	OAK ST / LAGUNA ST	-122.425891675136	37.7745985956747	
[2015-05-13 23:33:00]	Wednesday	NORTHERN	VAHNESS AV / GREE...	-122.42436302145	37.8004143219856	
[2015-05-13 23:30:00]	Wednesday	NORTHERN	1500 Block of LOM...	-122.42699532676599	37.80087263276921	
[2015-05-13 23:30:00]	Wednesday	PARK	100 Block of BROD...	-122.438737622757	37.771541172057795	

only showing top 5 rows

```
[4] data.printSchema()

root
 |-- Dates: string (nullable = true)
 |-- DayOfWeek: string (nullable = true)
 |-- PdDistrict: string (nullable = true)
 |-- Address: string (nullable = true)
 |-- X: string (nullable = true)
 |-- Y: string (nullable = true)
```

```
[5] from pyspark.sql.functions import col
data.groupBy("Address") \
    .count() \
    .orderBy(col("count").desc()) \
    .show()
```

The bottom of the notebook shows the Windows taskbar with the date 27-08-2022 and time 01:15.

Step-4: Data grouping by the Address and PdDistrict

The image displays two screenshots of a Google Colab notebook titled 'Untitled24.ipynb'. The interface includes a file explorer on the left with 'sample_data' and 'train.csv', a code editor in the center, and a terminal at the bottom. The top screenshot shows the execution of a PySpark SQL query to group data by 'Address' and sort by count. The bottom screenshot shows a similar query for 'PdDistrict' followed by a query to select specific columns.

Top Screenshot: Grouping by Address

```
[5] from pyspark.sql.functions import col
data.groupBy("Address") \
    .count() \
    .orderBy(col("count").desc()) \
    .show()
```

Address	count
800 Block of BRYA...	26533
800 Block of MARK...	6581
2000 Block of MIS...	5097
1000 Block of POT...	4063
900 Block of MARK...	3251
0 Block of TURK ST	3228
0 Block of 6TH ST	2884
300 Block of ELLI...	2793
400 Block of ELLI...	2590
16TH ST / MISSION ST	2504
1000 Block of MAR...	2489
1100 Block of MAR...	2319
2000 Block of MAR...	2168
100 Block of OFAR...	2140
700 Block of MARK...	2081
3200 Block of 20T...	2035
100 Block of 6TH ST	1887
500 Block of JOHN...	1824
TURK ST / TAYLOR ST	1810
200 Block of TURK ST	1800

only showing top 20 rows

Bottom Screenshot: Grouping by PdDistrict and Column Selection

```
[6] data.groupBy("PdDistrict") \
    .count() \
    .orderBy(col("count").desc()) \
    .show()
```

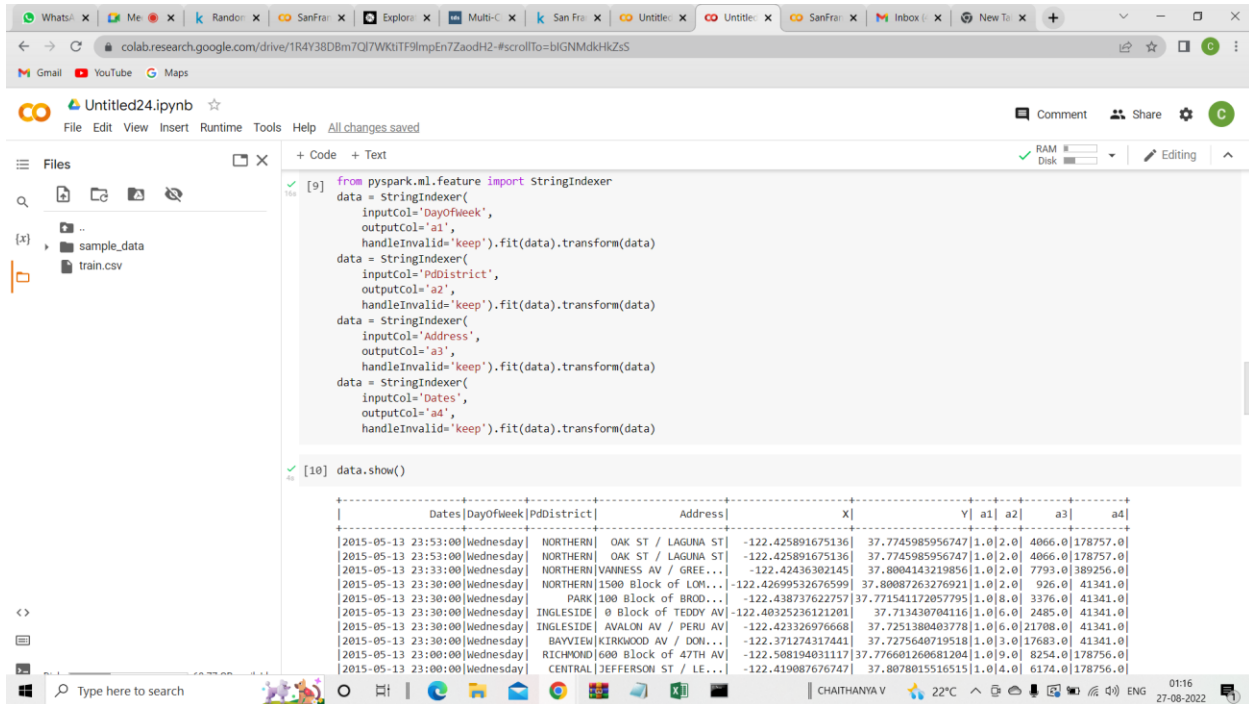
PdDistrict	count
SOUTHERN	157182
MISSION	119908
NORTHERN	105296
BAYVIEW	89431
CENTRAL	85460
TENDERLOIN	81809
INGLESIDE	78845
TARAVAL	65596
PARK	49313
RICHMOND	45209

```
[7] from pyspark.sql.functions import isnull, when, count
```

```
[8] data.select([count(when(isnull(c), c)).alias(c) for c in data.columns]).show()
```

Dates	DayOfWeek	PdDistrict	Address	X	Y
0	0	0	0	0	0

Step-5: Converting the string into numerical



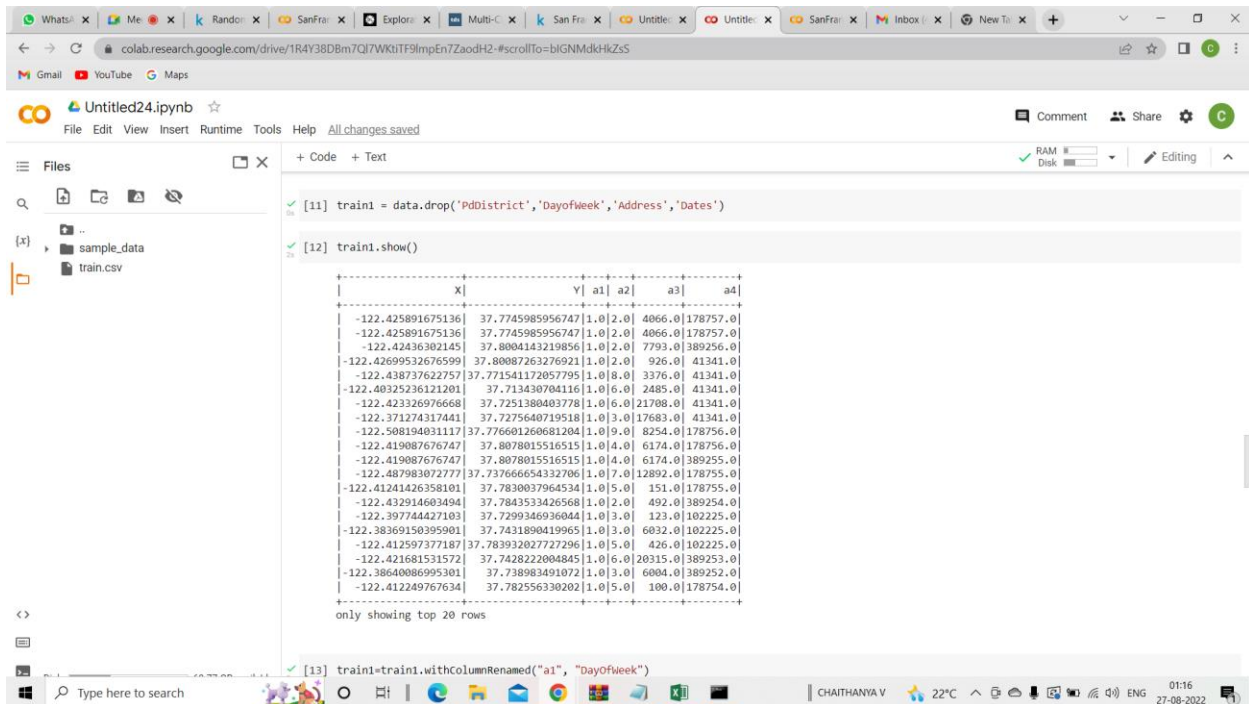
The screenshot shows a Google Colab notebook with the following code and output:

```
[9] from pyspark.ml.feature import StringIndexer
data = StringIndexer(
    inputCol='DayOfWeek',
    outputCol='a1',
    handleInvalid='keep').fit(data).transform(data)
data = StringIndexer(
    inputCol='PdDistrict',
    outputCol='a2',
    handleInvalid='keep').fit(data).transform(data)
data = StringIndexer(
    inputCol='Address',
    outputCol='a3',
    handleInvalid='keep').fit(data).transform(data)
data = StringIndexer(
    inputCol='Dates',
    outputCol='a4',
    handleInvalid='keep').fit(data).transform(data)
```

```
[10] data.show()
```

Dates	DayOfWeek	PdDistrict	Address	X	Y	a1	a2	a3	a4
2015-05-13 23:53:00	Wednesday	NORTHERN	OAK ST / LAGUNA ST	-122.425891675136	37.7745985956747	1.0	2.0	4066.0	178757.0
2015-05-13 23:53:00	Wednesday	NORTHERN	OAK ST / LAGUNA ST	-122.425891675136	37.7745985956747	1.0	2.0	4066.0	178757.0
2015-05-13 23:33:00	Wednesday	NORTHERN	VANNES AV / GREE...	-122.42436302145	37.8004143219856	1.0	2.0	7793.0	389256.0
2015-05-13 23:30:00	Wednesday	NORTHERN	1500 Block of LOM...	-122.42699532676599	37.80087263276921	1.0	2.0	926.0	41341.0
2015-05-13 23:30:00	Wednesday	PARK	100 Block of BROD...	-122.438737622757	37.771541172057795	1.0	8.0	3376.0	41341.0
2015-05-13 23:30:00	Wednesday	INGLESIDE	0 Block of TEDDY AV	-122.40325236121201	37.713430704116	1.0	6.0	2485.0	41341.0
2015-05-13 23:30:00	Wednesday	INGLESIDE	AVALON AV / PERU AV	-122.423326976668	37.7251380403778	1.0	6.0	21708.0	41341.0
2015-05-13 23:30:00	Wednesday	BAYVIEW	KIRKWOOD AV / DOW...	-122.371274317441	37.7275640719518	1.0	3.0	17683.0	41341.0
2015-05-13 23:00:00	Wednesday	RICHMOND	600 Block of 47TH AV	-122.508194031117	37.776601260681204	1.0	9.0	8254.0	178756.0
2015-05-13 23:00:00	Wednesday	CENTRAL	JEFFERSON ST / LE...	-122.419087676747	37.8078015516515	1.0	4.0	6174.0	178756.0

Step-6: Dropping the String rows and re-naming all the converted attributes



The screenshot shows a Google Colab notebook with the following code and output:

```
[11] train1 = data.drop('PdDistrict', 'DayOfWeek', 'Address', 'Dates')
```

```
[12] train1.show()
```

X	Y	a1	a2	a3	a4
-122.425891675136	37.7745985956747	1.0	2.0	4066.0	178757.0
-122.425891675136	37.7745985956747	1.0	2.0	4066.0	178757.0
-122.42436302145	37.8004143219856	1.0	2.0	7793.0	389256.0
-122.42699532676599	37.80087263276921	1.0	2.0	926.0	41341.0
-122.438737622757	37.771541172057795	1.0	8.0	3376.0	41341.0
-122.40325236121201	37.713430704116	1.0	6.0	2485.0	41341.0
-122.423326976668	37.7251380403778	1.0	6.0	21708.0	41341.0
-122.371274317441	37.7275640719518	1.0	3.0	17683.0	41341.0
-122.508194031117	37.776601260681204	1.0	9.0	8254.0	178756.0
-122.419087676747	37.8078015516515	1.0	4.0	6174.0	178756.0
-122.419087676747	37.8078015516515	1.0	4.0	6174.0	389255.0
-122.487983072777	37.7366684332706	1.0	7.0	12892.0	178755.0
-122.41241426358101	37.7830037964534	1.0	5.0	151.0	178755.0
-122.432914603494	37.7843533426568	1.0	2.0	492.0	389254.0
-122.397744427103	37.7299346936044	1.0	3.0	123.0	102225.0
-122.38369150395901	37.7431890419965	1.0	3.0	6032.0	102225.0
-122.412597377187	37.783932027727296	1.0	5.0	426.0	102225.0
-122.421681531572	37.7428222004845	1.0	6.0	20315.0	389253.0
-122.38640086995301	37.738983491072	1.0	3.0	6004.0	389252.0
-122.412249767634	37.782556330202	1.0	5.0	100.0	178754.0

only showing top 20 rows

```
[13] train1=train1.withColumnRenamed("a1", "DayOfWeek")
```

```

[13] train1=train1.withColumnRenamed("a1", "DayOfWeek")
train1=train1.withColumnRenamed("a2", "PdDistrict")
train1=train1.withColumnRenamed("a3", "Address")
train1=train1.withColumnRenamed("a4", "Dates")

train1.show()

```

X	Y	DayOfWeek	PdDistrict	Address	Dates
-122.425891675136	37.7745985956747	1.0	2.0	4066.0	178757.0
-122.425891675136	37.7745985956747	1.0	2.0	4066.0	178757.0
-122.42436302145	37.8004143219856	1.0	2.0	7793.0	389256.0
-122.42699532676599	37.80087263276921	1.0	2.0	926.0	41341.0
-122.438737622757	37.771541172057795	1.0	8.0	3376.0	41341.0
-122.40325236121201	37.7134380794116	1.0	6.0	2485.0	41341.0
-122.423326976668	37.7251380403778	1.0	6.0	21708.0	41341.0
-122.371274317441	37.7275640719518	1.0	3.0	17683.0	41341.0
-122.508194031117	37.776601260681204	1.0	9.0	8254.0	178756.0
-122.419087676747	37.8078015516515	1.0	4.0	6174.0	178756.0
-122.419087676747	37.8078015516515	1.0	4.0	6174.0	389255.0
-122.487983072777	37.73766654332706	1.0	7.0	12892.0	178755.0
-122.41241426358101	37.7830037964534	1.0	5.0	151.0	178755.0
-122.432914603494	37.7843533426568	1.0	2.0	492.0	389254.0
-122.397744427103	37.7299346936044	1.0	3.0	123.0	102225.0
-122.38369150395901	37.7431890419965	1.0	3.0	6032.0	102225.0
-122.412597377187	37.78393202772796	1.0	5.0	426.0	102225.0
-122.421681531572	37.7428222004845	1.0	6.0	20315.0	389253.0
-122.38640086995301	37.738983491072	1.0	3.0	6004.0	389252.0
-122.412249767634	37.782556330202	1.0	5.0	100.0	178754.0

only showing top 20 rows

3. Training the model using Random Forest and Naive bayes classifiers:

→Splitting the data into training and testing datasets.

Naïve bayes classifier: Fitting the preprocessed dataset in the model

```

[17] (trainingData, testData) = train1.randomSplit([0.7, 0.3], seed = 100)
print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))

Training Dataset Count: 614485
Test Dataset Count: 263564

[18] from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(smoothing=1)
model = nb.fit(trainingData)
predictions = model.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
.select("X", "Y", "probability", "PdDistrict", "DayOfWeek", "Address", "Dates", "prediction") \
.orderBy("probability", ascending=False) \
.show(n = 10, truncate = 30)

IllegalArgumentExcepton Traceback (most recent call last)
<ipython-input-18-85606011acd3> in <module>
      1 from pyspark.ml.classification import NaiveBayes
      2 nb = NaiveBayes(smoothing=1)
----> 3 model = nb.fit(trainingData)
      4 predictions = model.transform(testData)
      5 predictions.filter(predictions['prediction'] == 0) \

4 frames
/usr/local/lib/python3.7/dist-packages/pyspark/sql/utils.py in deco(*a, **kw)
    194 # Hide where the exception came from that shows a non-Pythonic
    195 # JVM exception message.
--> 196     raise converted from None
    197 else:
    198     raise

```



```
[18] from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(smoothing=1)
model = nb.fit(trainingData)
predictions = model.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
.select("X","Y","probability","PdDistrict","DayOfWeek","Address","Dates","prediction") \
.orderBy("probability", ascending=False) \
.show(n = 10, truncate = 30)

IllegalArgumentError
Traceback (most recent call last)
<ipython-input-18-85686011ac3> in <module>
1 from pyspark.ml.classification import NaiveBayes
2 nb = NaiveBayes(smoothing=1)
----> 3 model = nb.fit(trainingData)
4 predictions = model.transform(testData)
5 predictions.filter(predictions['prediction'] == 0) \

4 frames
/usr/local/lib/python3.7/dist-packages/pyspark/sql/utils.py in deco(*a, **kw)
194 # Hide where the exception came from that shows a non-Pythonic
195 # JVM exception message.
--> 196 raise converted from None
197 else:
198     raise

IllegalArgumentError: features does not exist. Available: X, Y, DayOfWeek, PdDistrict, Address, Dates
```

Random Forest classifier: Fitting the preprocessed dataset in the model

```
[19] from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(labelCol="label", \
                           featuresCol="features", \
                           numTrees = 100, \
                           maxDepth = 4, \
                           maxBins = 32)

# Train model with Training Data
rfModel = rf.fit(trainingData)
predictions = rfModel.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
.select("X","Y","probability","PdDistrict","DayOfWeek","Address","Dates","prediction") \
.orderBy("probability", ascending=False) \
.show(n = 10, truncate = 30)

IllegalArgumentError
Traceback (most recent call last)
<ipython-input-19-f771eeb22b3f> in <module>
6 maxBins = 32)
7 # Train model with Training Data
----> 8 rfModel = rf.fit(trainingData)
9 predictions = rfModel.transform(testData)
10 predictions.filter(predictions['prediction'] == 0) \

4 frames
/usr/local/lib/python3.7/dist-packages/pyspark/sql/utils.py in deco(*a, **kw)
194 # Hide where the exception came from that shows a non-Pythonic
195 # JVM exception message.
--> 196 raise converted from None
197 else:
198     raise

IllegalArgumentError: features does not exist. Available: X, Y, DayOfWeek, PdDistrict, Address, Dates
```