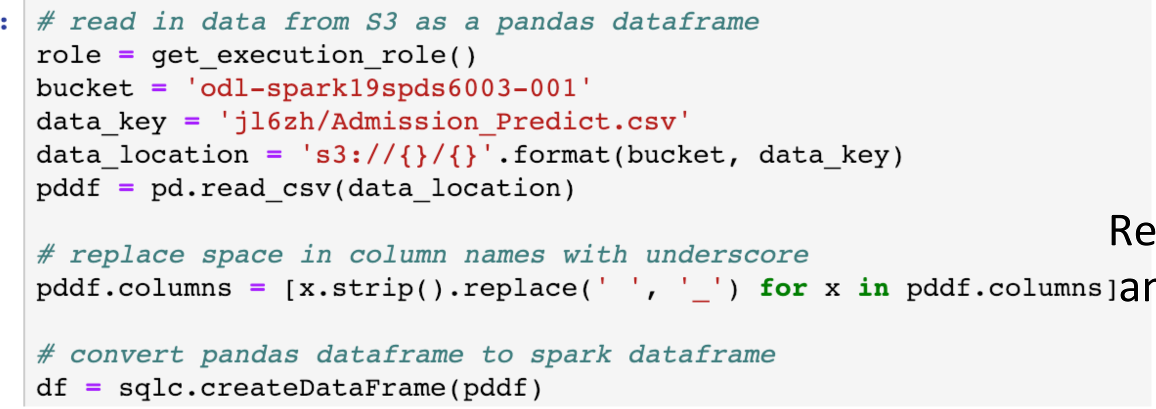
Jingyi Luo (jl6zh)

Motivation

- Universities post the basic requirements they want.
- Students are getting confused about what is the probability they could be admitted.
- This predicted result could give them a fair idea about the chance of admit.

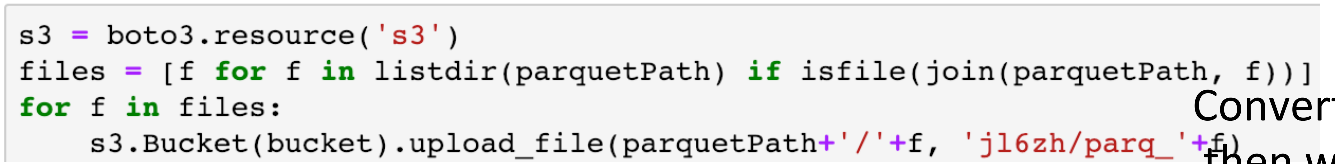
Code Snippets



```
# read in data from S3 as a pandas dataframe
role = get_execution_role()
bucket = 'odl-spark19spds6003-001'
data_key = 'jl6zh/Admission_Predict.csv'
data_location = 's3://{}/{}'.format(bucket, data_key)
pddf = pd.read_csv(data_location)

# replace space in column names with underscore
pddf.columns = [x.strip().replace(' ', '_') for x in pddf.columns]

# convert pandas dataframe to spark dataframe
df = sqlc.createDataFrame(pddf)
```

Read data from S3 as pandas dataframe, and convert it to spark dataframe

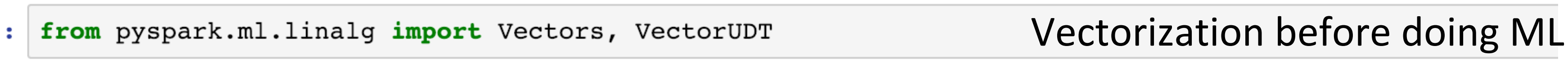
Write parquet to S3



```
s3 = boto3.resource('s3')
files = [f for f in listdir(parquetPath) if isfile(join(parquetPath, f))]
for f in files:
    s3.Bucket(bucket).upload_file(parquetPath+'/'+f, 'jl6zh/parq_'+f)
```

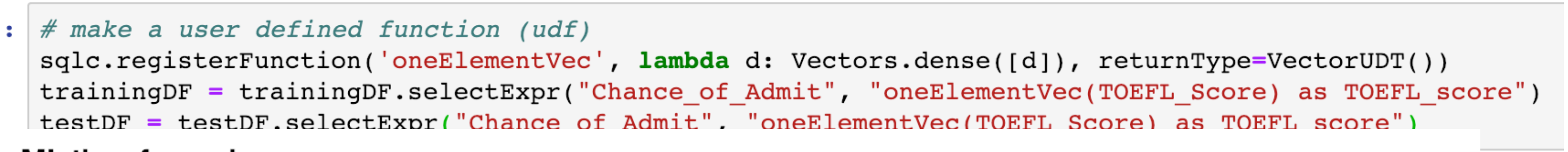
Convert spark dataframe to parquet, then write parquet to S3.

Vectorization - spark special sauce



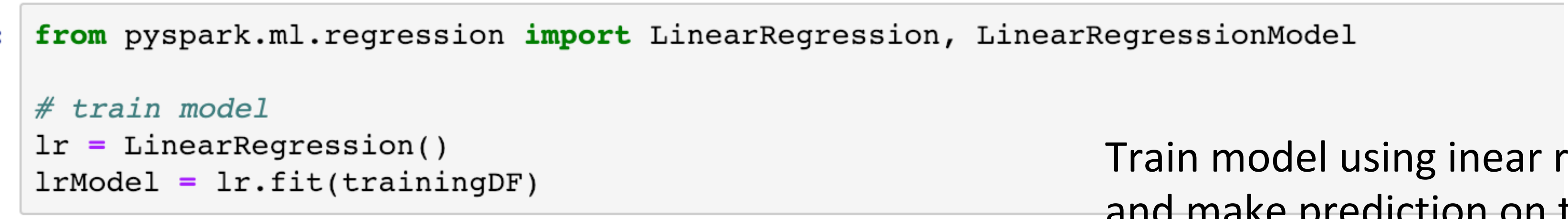
```
from pyspark.ml.linalg import Vectors, VectorUDT
```

Vectorization before doing ML



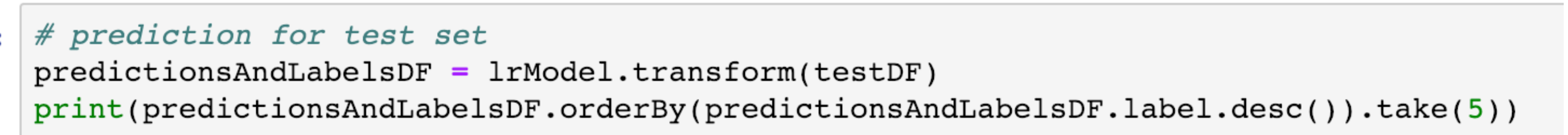
```
# make a user defined function (udf)
sqlc.registerFunction('oneElementVec', lambda d: Vectors.dense([d]), returnType=VectorUDT())
trainingDF = trainingDF.selectExpr("Chance_of_Admit", "oneElementVec(TOEFL_Score) as TOEFL_score")
testDF = testDF.selectExpr("Chance_of_Admit", "oneElementVec(TOEFL_Score) as TOEFL_score")
```

ML time for real



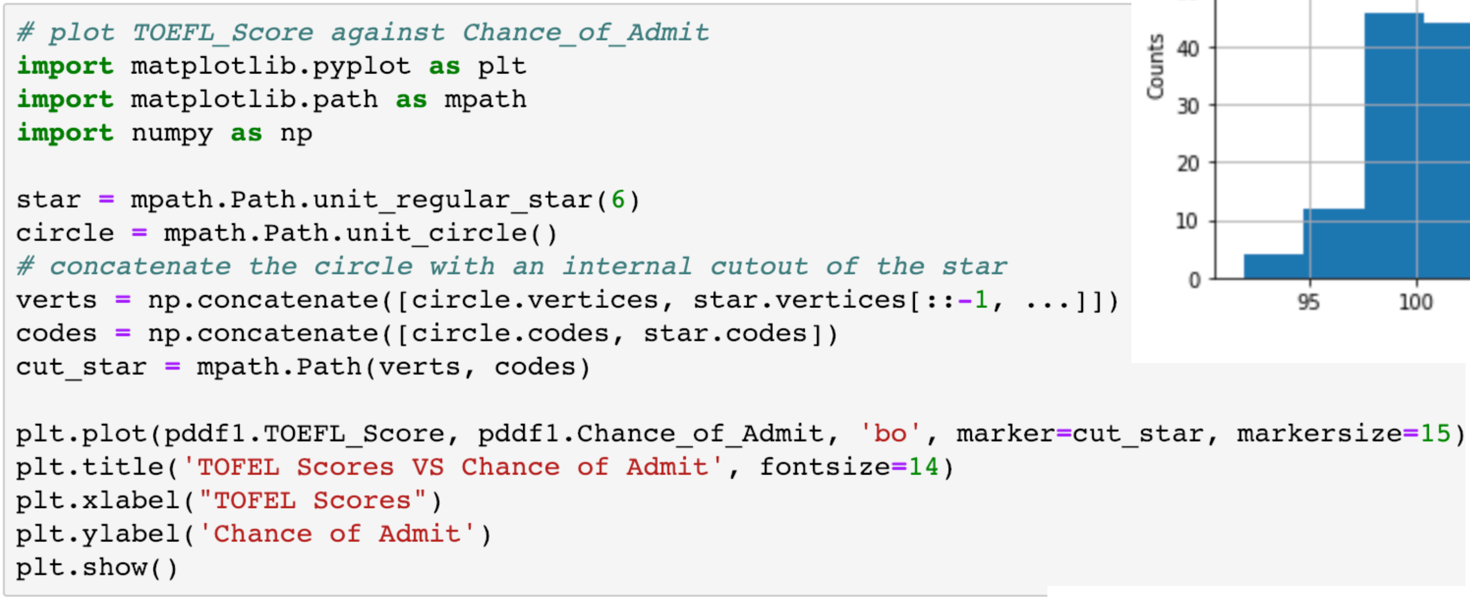
```
from pyspark.ml.regression import LinearRegression, LinearRegressionModel

# train model
lr = LinearRegression()
lrModel = lr.fit(trainingDF)
```

Train model using inear regression and make prediction on training set.

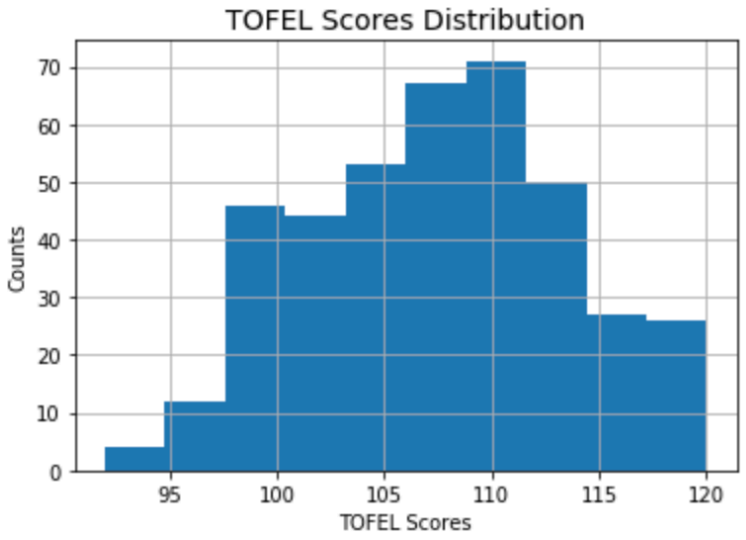


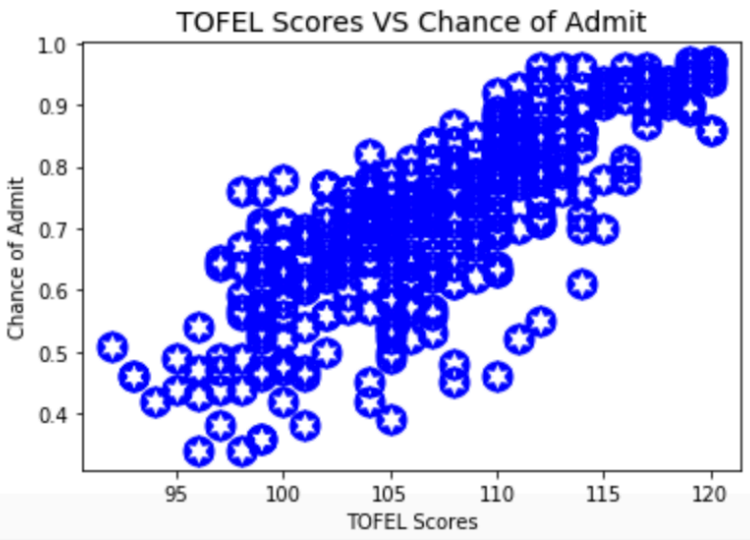
```
# prediction for test set
predictionsAndLabelsDF = lrModel.transform(testDF)
print(predictionsAndLabelsDF.orderBy(predictionsAndLabelsDF.label.desc()).take(5))
```

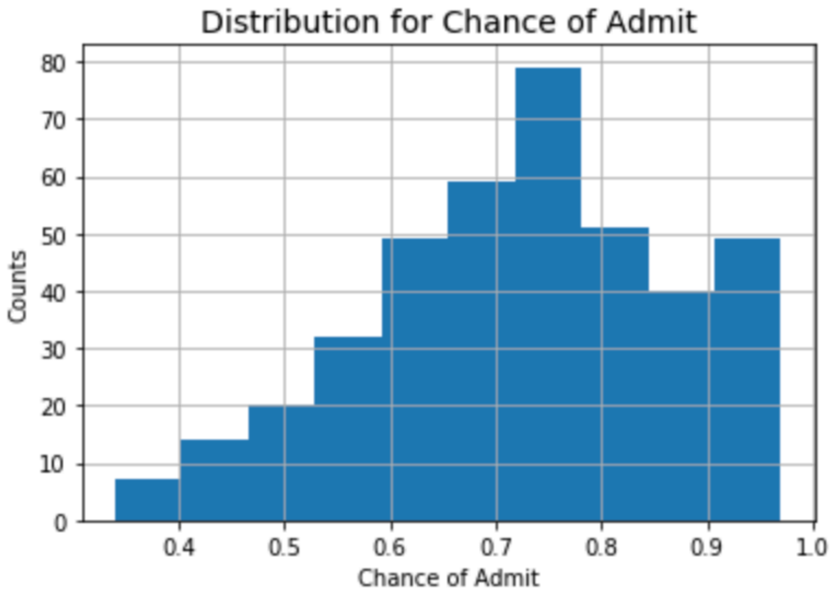
Visualization



```
# plot TOEFL_Score against Chance_of_Admit
import matplotlib.pyplot as plt
import matplotlib.path as mpath
import numpy as np

star = mpath.Path.unit_regular_star(6)
circle = mpath.Path.unit_circle()
# concatenate the circle with an internal cutout of the star
verts = np.concatenate([circle.vertices, star.vertices[::-1, ...]])
codes = np.concatenate([circle.codes, star.codes])
cut_star = mpath.Path(verts, codes)

plt.plot(pddf1.TOEFL_Score, pddf1.Chance_of_Admit, 'bo', marker=cut_star, markersize=15)
plt.title('TOFEL Scores VS Chance of Admit', fontsize=14)
plt.xlabel("TOFEL Scores")
plt.ylabel('Chance of Admit')
plt.show()
```


TOFEL Scores Distribution
Counts
TOFEL Scores


TOFEL Scores VS Chance of Admit
Chance of Admit
TOFEL Scores


Distribution for Chance of Admit
Counts
Chance of Admit