Sparkify

June 29, 2020

1 Sparkify Project Workspace - FE and Modelling Part

This workspace contains a tiny subset (128MB) of the full dataset available (12GB). Feel free to use this workspace to build your project, or to explore a smaller subset with Spark before deploying your cluster on the cloud. Instructions for setting up your Spark cluster is included in the last lesson of the Extracurricular Spark Course content.

You can follow the steps below to guide your data analysis and model building portion of this project.

```
In [99]: # import libraries
         from pyspark.sql import SparkSession

In [100]: # create a Spark session


          from pyspark.sql.functions import desc
          from pyspark.sql.functions import asc
          from pyspark.sql.functions import sum as Fsum
          from pyspark.sql.functions import avg, stddev, split, udf, isnull, first, col, format_
          from pyspark.sql.types import IntegerType, ArrayType, FloatType, DoubleType, Row, Date
          from pyspark.sql.functions import regexp_replace, col
          import pyspark.sql.functions as sf
          import pyspark.sql.types as st
          import pyspark.sql.functions as F
          from pyspark.ml.feature import CountVectorizer, IDF, Normalizer, PCA, RegexTokenizer,
          import datetime
          from pyspark.sql.functions import from_utc_timestamp, from_unixtime


          from pyspark.sql import Window
          import pandas as pd
          import numpy as np
          import seaborn as sns
          from matplotlib import pyplot as plt
          %matplotlib inline

          import re
          from pyspark.sql import functions as sF
```

```
        from pyspark.sql import types as sT

        from functools import reduce

        sns.set_style('whitegrid')
        from pyspark.ml.feature import VectorAssembler
        from pyspark.ml.feature import StandardScaler, VectorAssembler
```

```
In [101]: from pyspark.ml.feature import OneHotEncoderEstimator, StringIndexer, VectorAssembler
          from pyspark.ml import Pipeline
          from pyspark.ml.tuning import CrossValidator
          from pyspark.ml.evaluation import RegressionEvaluator
          from pyspark.sql.types import IntegerType, ArrayType, FloatType, DoubleType, Row, Date
          from pyspark.ml.linalg import DenseVector, SparseVector
          from pyspark.ml.classification import LogisticRegression, RandomForestClassifier, GBTC
          from pyspark.ml.evaluation import  MulticlassClassificationEvaluator
          from pyspark.ml.feature import CountVectorizer, IDF, Normalizer, PCA, RegexTokenizer,
          from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
          from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
In [102]: pd.set_option('display.max_columns', None)
          pd.set_option('display.expand_frame_repr', False)
          pd.set_option('max_colwidth', -1)
```

```
In [103]: from IPython.display import display, HTML

          display(HTML(data="""
          <style>
              div#notebook-container    { width: 95%; }
              div#menubar-container     { width: 65%; }
              div#maintoolbar-container { width: 99%; }
          </style>
          """))
```

```
<IPython.core.display.HTML object>
```

2 Load and Clean Dataset

In this workspace, the mini-dataset file is `mini_sparkify_event_data.json`. Load and clean the dataset, checking for invalid or missing data - for example, records without userids or sessionids.

3 Load and Clean Dataset

In this workspace, the mini-dataset file is `mini_sparkify_event_data.json`. Load and clean the dataset, checking for invalid or missing data - for example, records without userids or sessionids.

```
In [104]: # create a spark session
          def get_spark_session(master,appName):
              spark = SparkSession.builder.master(master).appName(appName).getOrCreate()
              return spark

In [105]: def load_dataset(filename):
              df_temp = spark.read.json(filename)
              return df_temp

In [106]: def feature_engineering_phase_1(df_local):
              df_local = df_local.withColumn('transaction_timestamp', from_unixtime(col('ts').ca
              df_local = df_local.withColumn('registration_timestamp', from_unixtime(col('regist
              df_local = df_local.withColumn('year', F.col('transaction_timestamp').cast('string
              df_local = df_local.withColumn('month', F.col('transaction_timestamp').cast('strin
              df_local = df_local.withColumn('day', F.col('transaction_timestamp').cast('string'
              df_local = df_local.withColumn('location', split(col('location'),',').getItem(1))
              df_local = df_local.withColumn('gender',F.when((col('gender')=='M'),1).otherwise(0
              #states = set([state[1].strip() for state in [x.split(',') for x in df_local.locat
              #states = set([state[1].strip() for state in [x.split(',') for x in np.array(df_lo
              # Define a user defined function
              #state = udf(lambda x: x.split(',')[1].strip())
              #df_local = df_local.withColumn("location", state(df.location))
              df_local = df_local.filter(df_local.userId != "")
              df_local = df_local.filter(col('userId').isNotNull())
              df_local = df_local.dropna(how = "any", subset = ["userId", "sessionId"])
              df_local = df_local.withColumn('churn_flag',F.when((col('page').isin(['Cancellatio
              return df_local

In [107]: def feature_engineering_phase_2(df_local):
              ex = '\(([^\)]*)\)'
              userAgents = [x for x  in np.array(df_local.select('userAgent').distinct().toPanda
              mapping = {'Compatible': 1,  'Ipad': 2,  'Iphone': 3, 'Macintosh': 4,  'Windows nt
              os_specific = udf(lambda x: mapping[re.findall(ex, x)[0].split(';')[0].capitalize(
              df_local = df_local.withColumn("os", os_specific(df_local.userAgent).cast('int'))
              df_local = df_local.withColumn('age', F.datediff(df.transaction_timestamp, df.regi
              df_local = df_local.drop('userAgent')
              df_local = df_local.drop('registration')
              for field in df_local.schema.fields:
                  if field.dataType==StringType():
                      df_local = df_local.withColumn(field.name, regexp_replace(field.name, '[^a
              df_level = df_local.orderBy('ts', ascending=False).groupBy('userId').agg(first('le
              df_status = df_local.orderBy('ts', ascending=False).groupBy('userId').agg(first('s
              df_local = df_local.join(df_level, on='userId')
              df_local = df_local.join(df_status, on='userId')
              df_local = df_local.drop('level')
              df_local = df_local.drop('status')
              df_local = df_local.drop('length')
              return df_local
```

```
In [108]: def feature_engineering_phase_3(df_new):
              df_new = df_new.groupby('userId','year','month','day').agg(F.countDistinct(df_new.
                                                                          F.max(df_new.location).
                                                                          F.max(df_new.last_level
                                                                          F.max(df_new.last_statu
                                                                          F.max(df_new.age).cast(
                                                                          F.max(df_new.os).cast('
                                                                          F.max(df_new.churn_flag
                                                                          F.count(df_new.song).ca
                                                                          F.max(df_new.itemInSess
                                                                          F.min(df_new.itemInSess
                                                                          F.avg(df_new.itemInSess
                                                                          ((F.max(df_new.transact
                                                                          ((F.min(df_new.transact
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.pa
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                          F.sum(F.when((df_new.so
                                                                       F.sum(F.when((df_new.so
                                                                       F.sum(F.when((df_new.so
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                       F.sum(F.when((df_new.ar
                                                                      )

        return df_new
```

```
In [109]: def feature_engineering_phase_4(df_new_local):
              df_new_local = df_new_local.groupby('userId').agg(   F.max(df_new_local.d_churn).c
                                                                   F.max(df_new_local.d_age).cas
                                                                   F.max(df_new_local.d_os).cast
                                                                   F.max(df_new_local.d_location
                                                                   F.max(df_new_local.d_last_lev
                                                                   F.max(df_new_local.d_last_sta
                                                                   F.sum(df_new_local.d_cnt_song
                                                                   F.sum(df_new_local.d_cnt_sess
                                                                   F.avg(df_new_local.d_cnt_sess
                                                                   F.max(df_new_local.d_cnt_sess
                                                                   F.max(df_new_local.d_max_item
                                                                   F.min(df_new_local.d_min_item
                                                                   F.avg(df_new_local.d_avg_item
                                                                   F.max(df_new_local.d_max_sess
                                                                   F.max(df_new_local.d_min_sess
                                                                   F.avg(df_new_local.d_cnt_roll
                                                                   F.avg(df_new_local.d_cnt_sett
                                                                   F.avg(df_new_local.d_cnt_down
                                                                   F.avg(df_new_local.d_cnt_next
                                                                   F.avg(df_new_local.d_cnt_erro
                                                                   F.avg(df_new_local.d_cnt_abou
                                                                   F.avg(df_new_local.d_cnt_upgr
                                                                   F.avg(df_new_local.d_cnt_home
                                                                   F.avg(df_new_local.d_cnt_logo
                                                                   F.avg(df_new_local.d_cnt_addt
                                                                   F.avg(df_new_local.d_cnt_thum
F.avg(df_new_local.d_cnt_thum
F.avg(df_new_local.d_cnt_save
F.avg(df_new_local.d_cnt_addf
F.avg(df_new_local.d_cnt_subm
F.avg(df_new_local.d_cnt_help
F.avg(df_new_local.d_cnt_subm
F.sum(df_new_local.d_cnt_roll
F.sum(df_new_local.d_cnt_sett
F.sum(df_new_local.d_cnt_down
F.sum(df_new_local.d_cnt_next
F.sum(df_new_local.d_cnt_erro
F.sum(df_new_local.d_cnt_abou
F.sum(df_new_local.d_cnt_upgr
F.sum(df_new_local.d_cnt_home
F.sum(df_new_local.d_cnt_logo
F.sum(df_new_local.d_cnt_addt
F.sum(df_new_local.d_cnt_thum
F.sum(df_new_local.d_cnt_thum
F.sum(df_new_local.d_cnt_save
F.sum(df_new_local.d_cnt_addf
F.sum(df_new_local.d_cnt_subm
F.sum(df_new_local.d_cnt_help
F.sum(df_new_local.d_cnt_subm
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_song
F.avg(df_new_local.d_cnt_king
F.avg(df_new_local.d_cnt_cold
F.avg(df_new_local.d_cnt_dwig
F.avg(df_new_local.d_cnt_flor
F.avg(df_new_local.d_cnt_theb
F.avg(df_new_local.d_cnt_bjrk
                                                    F.avg(df_new_local.d_cnt_just
                                                    F.avg(df_new_local.d_cnt_jack
                                                    F.avg(df_new_local.d_cnt_tayl
                                                    F.avg(df_new_local.d_cnt_harm
                                                    F.avg(df_new_local.d_cnt_alli
                                                    F.avg(df_new_local.d_cnt_guns
                                                    F.avg(df_new_local.d_cnt_trai
                                                    F.avg(df_new_local.d_cnt_emin
                                                    F.avg(df_new_local.d_cnt_oner
                                                          )
            df_new_local = df_new_local.drop('userId')
            return df_new_local
```

```
In [111]: spark = get_spark_session("local","Udacity - Sparkify")
          df=load_dataset('mini_sparkify_event_data.json')
```

```
In [113]: print('before cleaning {}'.format(df.count()))
          df = feature_engineering_phase_1(df)
          print('after cleaning {}'.format(df.count()))
```

```
before cleaning 286500
after cleaning 278154
```

First I cleaned the dataset, transformed some features to

```
In [114]: df = feature_engineering_phase_2(df)
```

```
In [115]: df = feature_engineering_phase_3(df)
          #df_dap = df.toPandas()
          #df_dap.head(10)
```

```
In [116]: df = feature_engineering_phase_4(df)
          #df_map = df.toPandas()
          #df_map.head(10)
```

```
In [117]: df.printSchema()
```

```
root
 |-- label: integer (nullable = true)
 |-- age: integer (nullable = true)
 |-- os: integer (nullable = true)
 |-- location: string (nullable = true)
 |-- 2m_last_level: string (nullable = true)
 |-- 2m_last_status: integer (nullable = true)
 |-- 2m_avg_song: long (nullable = true)
 |-- 2m_sum_session: long (nullable = true)
 |-- 2m_avg_session: double (nullable = true)
 |-- 2m_max_session: long (nullable = true)
|-- 2m_max_item_in_session: long (nullable = true)
|-- 2m_min_item_in_session: long (nullable = true)
|-- 2m_avg_item_in_session: integer (nullable = true)
|-- 2m_max_session_duration: integer (nullable = true)
|-- 2m_min_session_duration: integer (nullable = true)
|-- 2m_avg_page_rolladvert: double (nullable = true)
|-- 2m_avg_page_settings: double (nullable = true)
|-- 2m_avg_page_downgrade: double (nullable = true)
|-- 2m_avg_page_nextsong: double (nullable = true)
|-- 2m_avg_page_error: double (nullable = true)
|-- 2m_avg_page_about: double (nullable = true)
|-- 2m_avg_page_upgrade: double (nullable = true)
|-- 2m_avg_page_home: double (nullable = true)
|-- 2m_avg_page_logout: double (nullable = true)
|-- 2m_avg_page_addtoplaylist: double (nullable = true)
|-- 2m_avg_page_thumbsdown: double (nullable = true)
|-- 2m_avg_page_thumbsup: double (nullable = true)
|-- 2m_avg_page_savesettings: double (nullable = true)
|-- 2m_avg_page_addfriend: double (nullable = true)
|-- 2m_avg_page_submitupgrade: double (nullable = true)
|-- 2m_avg_page_help: double (nullable = true)
|-- 2m_avg_page_submitdowngrade: double (nullable = true)
|-- 2m_sum_page_rolladvert: long (nullable = true)
|-- 2m_sum_page_settings: long (nullable = true)
|-- 2m_sum_page_downgrade: long (nullable = true)
|-- 2m_sum_page_nextsong: long (nullable = true)
|-- 2m_sum_page_error: long (nullable = true)
|-- 2m_sum_page_about: long (nullable = true)
|-- 2m_sum_page_upgrade: long (nullable = true)
|-- 2m_sum_page_home: long (nullable = true)
|-- 2m_sum_page_logout: long (nullable = true)
|-- 2m_sum_page_addtoplaylist: long (nullable = true)
|-- 2m_sum_page_thumbsdown: long (nullable = true)
|-- 2m_sum_page_thumbsup: long (nullable = true)
|-- 2m_sum_page_savesettings: long (nullable = true)
|-- 2m_sum_page_addfriend: long (nullable = true)
|-- 2m_sum_page_submitupgrade: long (nullable = true)
|-- 2m_sum_page_help: long (nullable = true)
|-- 2m_sum_page_submitdowngrade: long (nullable = true)
|-- 2m_avg_song_youretheone: double (nullable = true)
|-- 2m_avg_song_revelry: double (nullable = true)
|-- 2m_avg_song_undo: double (nullable = true)
|-- 2m_avg_song_sehrkosmisch: double (nullable = true)
|-- 2m_avg_song_hornconcerto: double (nullable = true)
|-- 2m_avg_song_dogdaysareoverradio: double (nullable = true)
|-- 2m_avg_song_usesomebody: double (nullable = true)
|-- 2m_avg_song_secrets: double (nullable = true)
|-- 2m_avg_song_canada: double (nullable = true)
```

```
 |-- 2m_avg_song_sinceritetjalousie: double (nullable = true)
 |-- 2m_avg_song_aintmisbehavin: double (nullable = true)
 |-- 2m_avg_song_reprsente: double (nullable = true)
 |-- 2m_avg_song_lovestory: double (nullable = true)
 |-- 2m_avg_song_fireflies: double (nullable = true)
 |-- 2m_avg_song_catchyoubabysteve: double (nullable = true)
 |-- 2m_avg_song_heysoulsister: double (nullable = true)
 |-- 2m_avg_song_thegift: double (nullable = true)
 |-- 2m_avg_song_invalid: double (nullable = true)
 |-- 2m_avg_song_somebodytolove: double (nullable = true)
 |-- 2m_avg_artist_kingsofleon: double (nullable = true)
 |-- 2m_avg_artist_coldplay: double (nullable = true)
 |-- 2m_avg_artist_dwightyoakam: double (nullable = true)
 |-- 2m_avg_artist_florencethemachine: double (nullable = true)
 |-- 2m_avg_artist_theblackkeys: double (nullable = true)
 |-- 2m_avg_artist_bjrk: double (nullable = true)
 |-- 2m_avg_artist_justinbieber: double (nullable = true)
 |-- 2m_avg_artist_jackjohnson: double (nullable = true)
 |-- 2m_avg_artist_taylorswift: double (nullable = true)
 |-- 2m_avg_artist_harmonia: double (nullable = true)
 |-- 2m_avg_artist_allianceethnik: double (nullable = true)
 |-- 2m_avg_artist_gunsnroses: double (nullable = true)
 |-- 2m_avg_artist_train: double (nullable = true)
 |-- 2m_avg_artist_eminem: double (nullable = true)
 |-- 2m_avg_artist_onerepublic: double (nullable = true)
```

```
In [118]: #df.select([F.count(F.when(F.isnull(c), c)).alias(c) for c in df.columns]).toPandas()
```

4 Modeling

Split the full dataset into train, test, and validation sets. Test out several of the machine learning methods you learned. Evaluate the accuracy of the various models, tuning parameters as necessary. Determine your winning model based on test accuracy and report results on the validation set. Since the churned users are a fairly small subset, I suggest using F1 score as the metric to optimize.

```
In [121]: numeric_features = ['age','os','2m_last_status','2m_avg_song','2m_sum_session','2m_avg
                           '2m_min_item_in_session','2m_avg_item_in_session','2m_max_session_dura
                           '2m_avg_page_settings','2m_avg_page_downgrade','2m_avg_page_nextsong',
                           '2m_avg_page_home','2m_avg_page_logout','2m_avg_page_addtoplaylist','2
                           '2m_avg_page_addfriend','2m_avg_page_submitupgrade','2m_avg_page_help'
                           '2m_sum_page_downgrade','2m_sum_page_nextsong','2m_sum_page_error','2m
                           '2m_sum_page_addtoplaylist','2m_sum_page_thumbsdown','2m_sum_page_thum
                           '2m_sum_page_help','2m_sum_page_submitdowngrade','2m_avg_song_yourethe
                           '2m_avg_song_hornconcerto','2m_avg_song_dogdaysareoverradio','2m_avg_s
                           '2m_avg_song_aintmisbehavin','2m_avg_song_reprsente','2m_avg_song_love
```

```
                        '2m_avg_song_thegift','2m_avg_song_invalid','2m_avg_song_somebodytolov
                        '2m_avg_artist_florencethemachine','2m_avg_artist_theblackkeys','2m_av
                        '2m_avg_artist_taylorswift','2m_avg_artist_harmonia','2m_avg_artist_al
                        '2m_avg_artist_eminem','2m_avg_artist_onerepublic']

        indexer_location = StringIndexer(inputCol='location', outputCol='location_index')
        indexer_2m_last_level = StringIndexer(inputCol='2m_last_level', outputCol='level_index
        assembler = VectorAssembler(inputCols=numeric_features, outputCol='features')
        process_pipeline = Pipeline(stages=[indexer_location, indexer_2m_last_level, assembler
        modelling_dataframe = process_pipeline.fit(df).transform(df)

In [122]: train, test = modelling_dataframe.randomSplit([0.8, 0.2], seed=42)

In [123]: def fit_the_model(classifier):
              """
              fit and predict with training and test data
              :param classifier : Classifier Algorithm class
              :return classifer and prediction results
              """
              clf = classifier.fit(train)
              result = clf.transform(test)
              return clf, result

In [124]: def print_metrics(result):
              """
              :param result : prediction results which will be use to print metrics
              """
              evaluator= MulticlassClassificationEvaluator(predictionCol="prediction")
              print('Accuracy: {}'.format(evaluator.evaluate(result.select('label','prediction')
              print('F1 Score:{}'.format(evaluator.evaluate(result.select('label','prediction'),

In [125]: def print_feature_importance(clf, cols):
              a = {}
              feat_imp = clf.featureImportances
              for i in range(len(cols)):
                  a.update({cols[i]:feat_imp[i]})

              feat_importance = pd.DataFrame.from_dict(a, orient="index").reset_index()
              feat_importance.columns = ['feature', 'importance']
              feat_importance_top_20 = feat_importance.sort_values(by="importance", ascending=Fa
              return feat_importance_top_20

In [126]: def run_model_pipeline(model_algorithm):
              """
              :param model_algorithm : classifier algorithm class
              :return classifier, prediction result
              """
              clf, result = fit_the_model(model_algorithm)
              print_metrics(result)
              return clf,result
```

First model with randomforest with default params

```
In [127]: x,y=run_model_pipeline(RandomForestClassifier())
```

```
Accuracy: 0.7058823529411765
F1 Score:0.6288515406162466
Test F1 Score : 62.89%
```

First model with GradientBoosting with default params

```
In [128]: x,y=run_model_pipeline(GBTClassifier())
```

```
Accuracy: 0.7058823529411765
F1 Score:0.6591970121381886
Test F1 Score : 65.92%
```

First model with LogisticRegression with default params

```
In [129]: x,y=run_model_pipeline(LogisticRegression())
```

```
Accuracy: 0.7058823529411765
F1 Score:0.6954248366013073
Test F1 Score : 69.54%
```

According to default parameters, LogisticRegression is more good than the others based on f1 score

4.1 Tuning Model

```
In [132]: indexer_location = StringIndexer(inputCol='location', outputCol='location_index')
          indexer_2m_last_level = StringIndexer(inputCol='2m_last_level', outputCol='level_index
          assembler = VectorAssembler(inputCols=numeric_features, outputCol='features')
```

```
In [163]: lr =  LogisticRegression(maxIter=10, regParam=0.0, elasticNetParam=0)
          indexer_location = StringIndexer(inputCol='location', outputCol='location_index')
          indexer_2m_last_level = StringIndexer(inputCol='2m_last_level', outputCol='level_index
          assembler = VectorAssembler(inputCols=numeric_features, outputCol='features')
          pipeline_tune_lr = Pipeline(stages=[indexer_location, indexer_2m_last_level, assembler
```

```
In [167]: numTrees=[20,60]
          maxDepth=[10,30]
```

```
In [179]: def tune_model(classifier_model_alg, numTrees_param, maxDepth_param, metricName_param,
              paramGrid = ParamGridBuilder().addGrid(classifier_model_alg.numTrees, numTrees_par
              crossval = CrossValidator(estimator = Pipeline(stages=[classifier_model_alg]), est
              cross_validation_model = crossval.fit(train)
              prediction_results = cross_validation_model.transform(test)
              return prediction_results
```

tuning the randomforest alg

```
In [168]: predictions = tune_model(RandomForestClassifier(), [20,60], [10,30], 'f1', 3 )
```

```
In [169]: print_metrics(predictions)
```

```
Accuracy: 0.7647058823529411
F1 Score:0.7030812324929971
Test F1 Score : 70.31%
```

tuning the randomforest alg

```
In [181]: pred_results = tune_model(RandomForestClassifier(), [20,70], [5,30], 'f1', 4 )
```

```
In [182]: print_metrics(pred_results)
```

```
Accuracy: 0.7352941176470589
F1 Score:0.6479031804109204
Test F1 Score : 64.79%
```

```
In [183]: pred_results = tune_model(RandomForestClassifier(), [20,80], [10,25], 'f1', 5 )
```

```
In [184]: print_metrics(pred_results)
```

```
Accuracy: 0.7058823529411765
F1 Score:0.5841784989858012
Test F1 Score : 58.42%
```

tuning the randomforest alg

```
In [188]: predictions = tune_model(RandomForestClassifier(), [20,75], [10,30], 'f1', 3 )
```

```
In [189]: print_metrics(predictions)
```

```
Accuracy: 0.7647058823529411
F1 Score:0.7030812324929971
Test F1 Score : 70.31%
```

```
In [204]: indexer = StringIndexer(inputCol='label', outputCol='label')
          assembler = VectorAssembler(inputCols=numeric_features, outputCol='features')
```

```
In [207]: def tune_model_lr(classifier_model_alg, maxIter_param,regParam_param, elasticNetParam_
              pipeline_lr_tuned = Pipeline(stages=[assembler, indexer, lr])
              paramGrid_lr_tuned = ParamGridBuilder().addGrid(classifier_model_alg.maxIter, maxI

              # Cross validator for above grid parameters
              crossval_lr_tuned = CrossValidator(estimator=classifier_model_alg, estimatorParamM
              crossval_lr_model_tuned = crossval_lr_tuned.fit(train)
              prediction_results_tuned = crossval_lr_model_tuned.transform(test)
              return prediction_results_tuned
```

tuning the LogisticRegression alg

```
In [208]: predictions_lr = tune_model_lr(LogisticRegression(), [10, 20], [0.0, 0.1],[0.0, 0.5],3
```

```
In [209]: print_metrics(predictions_lr)
```

```
Accuracy: 0.7058823529411765
F1 Score:0.5841784989858012
Test F1 Score : 58.42%
```

tuning the LogisticRegression alg

```
In [210]: predictions_lr2 = tune_model_lr(LogisticRegression(), [5, 25], [0.1, 0.3],[0.0, 0.8],4
```

```
In [211]: print_metrics(predictions_lr2)
```

```
Accuracy: 0.7647058823529411
F1 Score:0.7030812324929971
Test F1 Score : 70.31%
```

```
In [213]: predictions_lr3 = tune_model_lr(LogisticRegression(), [5, 30], [0.1, 0.5],[0.0, 0.9],4
          print_metrics(predictions_lr3)
```

```
Accuracy: 0.7647058823529411
F1 Score:0.7030812324929971
Test F1 Score : 70.31%
```

tuning the LogisticRegression alg

```
In [214]: predictions_lr4 = tune_model_lr(LogisticRegression(), [10, 60], [0.1, 0.5],[0.0, 1.0],
          print_metrics(predictions_lr4)
```

```
Accuracy: 0.7647058823529411
F1 Score:0.7030812324929971
Test F1 Score : 70.31%
```

As you see, LogisticRegression was the best model for this work

5 Final Steps

Clean up your code, adding comments and renaming variables to make the code easier to read and maintain. Refer to the Spark Project Overview page and Data Scientist Capstone Project Rubric to make sure you are including all components of the capstone project and meet all expectations. Remember, this includes thorough documentation in a README file in a Github repository, as well as a web app or blog post.