**PROJECT – US Election**

**PROBLEM STATEMENT:**

In the US Primary Election 2016, Hillary Clinton was nominated over Bernie Sanders from Democrats and on the other hand, Donald Trump was nominated from Republican Party to contest for the presidential position. As an analyst, you have been tasked to understand different factors that led to the winning of Hillary Clinton and Donald Trump in the primary elections based on demographic features to plan their next initiatives and campaigns.

**STRATEGY:**

1. Storage -> The US Election data set I copied from local filesystem to HDFS. Where the HDFS is used as storage system. But here I used local file system.

We can do it by using mini hdfs cluster. I will be mentioning the mini hdfs cluster class in end of document.

1. Processing Framework -> Where Spark is used as processing framework and importing all requires libraries
2. Spark SQl -> Spark SQL is used for querying the data set and give results in tabular format and filter some of the records.
3. Spark Mlib -> Spark Mlib is used for using machine learning algorithms on data set for further analysis.
4. Zeppelin -> where it is framework used to visualize the intermediate and final results.

**CODE:**

import org.apache.spark.{SparkConf, SparkContext}

import org.apache.spark.sql.{SQLContext}

import org.apache.spark.sql.types.{DoubleType, IntegerType, StringType, StructField,StructType}

import org.apache.spark.ml.feature.VectorAssembler

import org.apache.spark.ml.clustering.KMeans

val master\_url="local[2]"

val temp\_dir="C:\\Users\\MUKHESH\\OneDrive\\Documents\\Temp\_Dir"

val app\_name="US\_Election"

var sc:SparkContext=null

// var spark:SparkSession=null

val conf=new SparkConf().setAppName(app\_name).setMaster(master\_url).set("spark.sql.shuffle.partitions","2").set("saprk.executor.memory","2g")

sc=SparkContext.getOrCreate(conf)

// spark=SparkSession.builder().appName(app\_name).master(master\_url).config("spark.sql.warehouse.dir",temp\_dir).getOrCreate()

val spark=new SQLContext(sc)

import spark.implicits.\_

val schema1=StructType(Array(

StructField("state",StringType,true),

StructField("state\_abbreviation",StringType,true),

StructField("county",StringType,true),

StructField("fips",StringType,true),

StructField("party",StringType,true),

StructField("candidate",StringType,true),

StructField("votes",IntegerType,true),

StructField("fraction\_votes",DoubleType,true)))

val schema = StructType(Array( StructField("fips", StringType, true), StructField("area\_name",

StringType, true), StructField("state\_abbreviation", StringType, true),

StructField("Population\_2014", IntegerType, true), StructField("Population\_2010\_Apr1",

IntegerType, true), StructField("Change\_in\_Population\_percent", DoubleType, true),

StructField("Population\_2010", IntegerType, true), StructField ("Persons\_under\_5",

DoubleType, true), StructField("Persons\_under\_18", DoubleType, true), StructField

("Persons\_65\_years\_over", DoubleType, true), StructField("Female\_persons\_percent",

DoubleType, true), StructField("White\_alone", DoubleType, true),

StructField("Black\_or\_African\_American\_alone", DoubleType, true),

StructField("American\_Indian\_and\_Alaska\_Native\_alone", DoubleType, true),

StructField("Asian\_alone", DoubleType, true),

StructField("Native\_Hawaiian\_and\_Other\_Pacific\_Islander\_alone", DoubleType, true),

StructField("Two\_or\_More\_Races", DoubleType, true), StructField("Hispanic\_or\_Latino",

DoubleType, true), StructField("White\_alone\_not\_Hispanic\_or\_Latino", DoubleType, true),

StructField("Living\_in\_same\_house\_1\_year\_&\_over", DoubleType, true),

StructField("Foreign\_born\_persons", DoubleType, true),

StructField("Language\_other\_than\_English\_spoken\_at\_home", DoubleType, true),

StructField("High\_school\_graduate\_or\_higher", DoubleType, true),

StructField("Bachelor\_degree\_or\_higher", DoubleType, true), StructField("Veterans",

IntegerType, true), StructField("Mean\_travel\_time\_to\_work", DoubleType, true),

StructField("Housing\_units", IntegerType, true), StructField ("Homeownership\_rate",

DoubleType, true), StructField("Housing\_units\_in\_multi\_unit\_structures", DoubleType,

true), StructField("Median\_value\_of\_owner\_occupied\_housing\_units", IntegerType, true),

StructField("Households", IntegerType, true), StructField("Persons\_per\_household",

DoubleType, true), StructField("Per\_capita\_money\_income", IntegerType, true),

StructField("Median\_household\_income", IntegerType, true),

StructField("Persons\_below\_poverty\_level", DoubleType, true),

StructField("Private\_nonfarm\_establishments", IntegerType, true),

StructField("Private\_nonfarm\_employment", IntegerType, true),

StructField("Private\_nonfarm\_employment\_percentage\_change", DoubleType, true),

StructField ("Nonemployer\_establishments", IntegerType, true),

StructField("Total\_number\_of\_firms", IntegerType, true), StructField("Black\_owned\_firms",

DoubleType, true), StructField("American\_Indian\_and\_Alaska\_Native\_owned\_firms",

DoubleType, true), StructField("Asian\_owned\_firms", DoubleType, true),

StructField("Native\_Hawaiian\_and\_Other\_Pacific\_Islander\_owned\_firms", DoubleType,

true), StructField ("Hispanic\_owned\_firms", DoubleType, true),

StructField("Women\_owned\_firms", DoubleType, true),

StructField("Manufacturers\_shipments", DoubleType, true),

StructField("Merchant\_wholesaler\_sales", DoubleType, true), StructField("Retail\_sales",

DoubleType, true), StructField("Retail\_sales\_per\_capita", IntegerType, true),

StructField("Accommodation\_and\_food\_services\_sales", IntegerType, true),

StructField("Building\_permits", IntegerType, true),

StructField("Land\_area\_in\_square\_miles", DoubleType, true),

StructField("Population\_per\_square\_mile", DoubleType, true)))

val results=spark.read.format("csv").option("header","true").schema(schema1).load("D:\\Edureka\\Edureka share folders\\US\_Election\_Dataset\\primary\_results.csv")

val county\_facts=spark.read.format("csv").option("header","true").schema(schema).load("D:\\Edureka\\Edureka share folders\\US\_Election\_Dataset\\county\_facts.csv")

val democrats=results.filter($"party"==="Democrat")

val republican=results.filter($"party"==="Republican")

democrats.registerTempTable("democrats")

republican.registerTempTable("republicans")

val democrat\_results=spark.sql("SELECT a.state as state,a.candidate as candidate,a.fips as fips,a.county as county,a.fraction\_votes as fraction\_votes from (select fips,max(fraction\_votes) as fraction\_votes from democrats group by fips) b join democrats a on a.fips=b.fips and a.fraction\_votes=b.fraction\_votes")

val republican\_results=spark.sql("SELECT a.state as state,a.candidate as candidate,a.fips as fips,a.county as county,a.fraction\_votes as fraction\_votes from (select fips,max(fraction\_votes) as fraction\_votes from republicans group by fips) b join republicans a on a.fips=b.fips and a.fraction\_votes=b.fraction\_votes")

democrat\_results.registerTempTable("dc\_result")

republican\_results.registerTempTable("rp\_results")

county\_facts.registerTempTable("facts")

//let us few common columns which has impact on analysis

val de\_result=spark.sql("SELECT facts.fips as fips, dc\_result.state as state, facts.state\_abbreviation as state\_abbreviation, area\_name, candidate, Persons\_65\_years\_over,Female\_persons\_percent, White\_alone, Black\_or\_African\_American\_alone, Asian\_alone,Hispanic\_or\_Latino, Foreign\_born\_persons, Language\_other\_than\_English\_spoken\_at\_home,Bachelor\_degree\_or\_higher, Veterans, Homeownership\_rate, Median\_household\_income,Persons\_below\_poverty\_level, Population\_per\_square\_mile FROM facts INNER JOIN dc\_result ON CAST(facts.fips AS INT) = CAST(dc\_result.fips AS INT)")

val re\_result=spark.sql("SELECT facts.fips as fips, rp\_results.state as state, facts.state\_abbreviation as state\_abbreviation, area\_name, candidate, Persons\_65\_years\_over,Female\_persons\_percent, White\_alone, Black\_or\_African\_American\_alone, Asian\_alone,Hispanic\_or\_Latino, Foreign\_born\_persons, Language\_other\_than\_English\_spoken\_at\_home,Bachelor\_degree\_or\_higher, Veterans, Homeownership\_rate, Median\_household\_income,Persons\_below\_poverty\_level, Population\_per\_square\_mile FROM facts INNER JOIN rp\_results ON CAST(facts.fips AS INT) = CAST(rp\_results.fips AS INT)")

//in democrats the most states is won by Hillary and Ben but martin did'nt won any thing

val hillary=de\_result.filter($"candidate"==="Hillary Clinton")

val ben=de\_result.filter($"candidate"==="Bernie Sanders")

//creating another variables

val wh=hillary.withColumn("wh",lit(1)).withColumn("wb",lit(0))

val wb=ben.withColumn("wh",lit(0)).withColumn("wb",lit(1))

//where it comes to republican there are five players giving competition

val bc = re\_result.filter($"candidate" === "Ben Carson")

val dt = re\_result.filter($"candidate" === "Donald Trump")

val jk = re\_result.filter($"candidate" === "John Kasich")

val mr = re\_result.filter($"candidate" === "Marco Rubio")

val tc = re\_result.filter($"candidate" === "Ted Cruz")

//creating another variables

val wbc = bc.withColumn("w\_bc", lit(1)).withColumn("w\_dt", lit(0)).withColumn("w\_jk",lit(0)).withColumn("w\_mr", lit(0)).withColumn("w\_tc", lit(0))

val wdt = dt.withColumn("w\_bc", lit(0)).withColumn("w\_dt", lit(1)).withColumn("w\_jk",lit(0)).withColumn("w\_mr", lit(0)).withColumn("w\_tc", lit(0))

val wjk = jk.withColumn("w\_bc", lit(0)).withColumn("w\_dt", lit(0)).withColumn("w\_jk",lit(1)).withColumn("w\_mr", lit(0)).withColumn("w\_tc", lit(0))

val wmr = mr.withColumn("w\_bc", lit(0)).withColumn("w\_dt", lit(0)).withColumn("w\_jk",lit(0)).withColumn("w\_mr", lit(1)).withColumn("w\_tc", lit(0))

val wtc = tc.withColumn("w\_bc", lit(0)).withColumn("w\_dt", lit(0)).withColumn("w\_jk",lit(0)).withColumn("w\_mr", lit(0)).withColumn("w\_tc", lit(1))

//creating the tables

wh.createOrReplaceTempView("wh\_d")

wb.createOrReplaceTempView("wb\_d")

//creating the tables

wbc.createOrReplaceTempView("wbc\_r")

wdt.createOrReplaceTempView("wdt\_r")

wjk.createOrReplaceTempView("wjk\_r")

wmr.createOrReplaceTempView("wmr\_r")

wtc.createOrReplaceTempView("wtc\_r")

//combbining all created tables.

val result\_d=spark.sql("SELECT \* FROM wh\_d UNION ALL SELECT \* FROM wb\_d")

val result\_r=spark.sql("SELECT \* FROM wbc\_r UNION ALL SELECT \* FROM wdt\_r UNION ALL SELECT \* FROM wjk\_r UNION ALL SELECT \* FROM wmr\_r UNION ALL SELECT \* FROM wtc\_r")

result\_d.registerTempTable("result\_d")

result\_r.registerTempTable("result\_r")

//creating the features for machine learning

val features\_col\_r=Array("Persons\_65\_years\_over", "Female\_persons\_percent",

"White\_alone","Black\_or\_African\_American\_alone", "Asian\_alone", "Hispanic\_or\_Latino",

"Foreign\_born\_persons", "Language\_other\_than\_English\_spoken\_at\_home",

"Bachelor\_degree\_or\_higher", "Veterans", "Homeownership\_rate",

"Median\_household\_income", "Persons\_below\_poverty\_level",

"Population\_per\_square\_mile", "w\_bc", "w\_dt", "w\_jk", "w\_mr", "w\_tc")

val features\_col\_d=Array("Persons\_65\_years\_over", "Female\_persons\_percent",

"White\_alone","Black\_or\_African\_American\_alone", "Asian\_alone", "Hispanic\_or\_Latino",

"Foreign\_born\_persons", "Language\_other\_than\_English\_spoken\_at\_home",

"Bachelor\_degree\_or\_higher", "Veterans", "Homeownership\_rate",

"Median\_household\_income", "Persons\_below\_poverty\_level",

"Population\_per\_square\_mile","wh","wb")

val rows\_r=new VectorAssembler().setInputCols(features\_col\_r).setOutputCol("features").transform(result\_r)

val rows\_d=new VectorAssembler().setInputCols(features\_col\_d).setOutputCol("features").transform(result\_d)

//kmeans for democrats

val kmeans\_d=new KMeans().setK(4).setFeaturesCol("features").setPredictionCol("predictions")

val model\_d=kmeans\_d.fit(rows\_d)

model\_d.clusterCenters.foreach(println)

val categories\_d=model\_d.transform(rows\_d)

categories\_d.registerTempTable("categories\_d")

val kmeans\_r=new KMeans().setK(6).setFeaturesCol("features").setPredictionCol("predictions")

val model\_r=kmeans\_r.fit(rows\_r)

model\_r.clusterCenters.foreach(println)

//republicans kmeans transform data storing into table

val categories\_r=model\_r.transform(rows\_r)

categories\_r.registerTempTable("categories\_r")

**Analysis:**

Where Sanders has a popularity in foreign born who is Asian mostly where if Donald trump uses this and make initiative for Asian people for Indian and Chinese people donald trump has more probability in winning.

Where Hillary has good popularity in blacks but Donald trump did’nt where Hillary cannot importance to black because she might lose the support of white people. Need to concentrate on trump allegations.

The following below are few of the charts which bring me into the conclusion.





















