package com.twitter.timelines.prediction.features.client\_log\_event

import com.twitter.dal.personal\_data.thriftjava.PersonalDataType.\_

import com.twitter.ml.api.Feature

import com.twitter.ml.api.Feature.Binary

import com.twitter.ml.api.Feature.Continuous

import com.twitter.ml.api.Feature.Discrete

import scala.collection.JavaConverters.\_

import com.twitter.timelineservice.suggests.logging.candidate\_tweet\_source\_id.thriftscala.CandidateTweetSourceId

object ClientLogEventDataRecordFeatures {

val HasConsumerVideo = new Binary(

"client\_log\_event.tweet.has\_consumer\_video",

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val PhotoCount = new Continuous(

"client\_log\_event.tweet.photo\_count",

Set(CountOfPrivateTweetEntitiesAndMetadata, CountOfPublicTweetEntitiesAndMetadata).asJava)

val HasImage = new Binary(

"client\_log\_event.tweet.has\_image",

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val IsReply =

new Binary("client\_log\_event.tweet.is\_reply", Set(PublicReplies, PrivateReplies).asJava)

val IsRetweet =

new Binary("client\_log\_event.tweet.is\_retweet", Set(PublicRetweets, PrivateRetweets).asJava)

val IsPromoted =

new Binary(

"client\_log\_event.tweet.is\_promoted",

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val HasVisibleLink = new Binary(

"client\_log\_event.tweet.has\_visible\_link",

Set(UrlFoundFlag, PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val HasHashtag = new Binary(

"client\_log\_event.tweet.has\_hashtag",

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val FromMutualFollow = new Binary("client\_log\_event.tweet.from\_mutual\_follow")

val IsInNetwork = new Binary("client\_log\_event.tweet.is\_in\_network")

val IsNotInNetwork = new Binary("client\_log\_event.tweet.is\_not\_in\_network")

val FromRecap = new Binary("client\_log\_event.tweet.from\_recap")

val FromRecycled = new Binary("client\_log\_event.tweet.from\_recycled")

val FromActivity = new Binary("client\_log\_event.tweet.from\_activity")

val FromSimcluster = new Binary("client\_log\_event.tweet.from\_simcluster")

val FromErg = new Binary("client\_log\_event.tweet.from\_erg")

val FromCroon = new Binary("client\_log\_event.tweet.from\_croon")

val FromList = new Binary("client\_log\_event.tweet.from\_list")

val FromRecTopic = new Binary("client\_log\_event.tweet.from\_rec\_topic")

val InjectedPosition = new Discrete("client\_log\_event.tweet.injectedPosition")

val TextOnly = new Binary("client\_log\_event.tweet.text\_only")

val HasLikedBySocialContext = new Binary("client\_log\_event.tweet.has\_liked\_by\_social\_context")

val HasFollowedBySocialContext = new Binary(

"client\_log\_event.tweet.has\_followed\_by\_social\_context")

val HasTopicSocialContext = new Binary("client\_log\_event.tweet.has\_topic\_social\_context")

val IsFollowedTopicTweet = new Binary("client\_log\_event.tweet.is\_followed\_topic\_tweet")

val IsRecommendedTopicTweet = new Binary("client\_log\_event.tweet.is\_recommended\_topic\_tweet")

val IsTweetAgeLessThan15Seconds = new Binary(

"client\_log\_event.tweet.tweet\_age\_less\_than\_15\_seconds")

val IsTweetAgeLessThanOrEqualTo30Minutes = new Binary(

"client\_log\_event.tweet.tweet\_age\_lte\_30\_minutes")

val IsTweetAgeLessThanOrEqualTo1Hour = new Binary("client\_log\_event.tweet.tweet\_age\_lte\_1\_hour")

val IsTweetAgeLessThanOrEqualTo6Hours = new Binary("client\_log\_event.tweet.tweet\_age\_lte\_6\_hours")

val IsTweetAgeLessThanOrEqualTo12Hours = new Binary(

"client\_log\_event.tweet.tweet\_age\_lte\_12\_hours")

val IsTweetAgeGreaterThanOrEqualTo24Hours = new Binary(

"client\_log\_event.tweet.tweet\_age\_gte\_24\_hours")

val HasGreaterThanOrEqualTo100Favs = new Binary("client\_log\_event.tweet.has\_gte\_100\_favs")

val HasGreaterThanOrEqualTo1KFavs = new Binary("client\_log\_event.tweet.has\_gte\_1k\_favs")

val HasGreaterThanOrEqualTo10KFavs = new Binary("client\_log\_event.tweet.has\_gte\_10k\_favs")

val HasGreaterThanOrEqualTo100KFavs = new Binary("client\_log\_event.tweet.has\_gte\_100k\_favs")

val HasGreaterThanOrEqualTo10Retweets = new Binary("client\_log\_event.tweet.has\_gte\_10\_retweets")

val HasGreaterThanOrEqualTo100Retweets = new Binary("client\_log\_event.tweet.has\_gte\_100\_retweets")

val HasGreaterThanOrEqualTo1KRetweets = new Binary("client\_log\_event.tweet.has\_gte\_1k\_retweets")

val TweetTypeToFeatureMap: Map[String, Binary] = Map(

"link" -> HasVisibleLink,

"hashtag" -> HasHashtag,

"mutual\_follow" -> FromMutualFollow,

"in\_network" -> IsInNetwork,

"text\_only" -> TextOnly,

"has\_liked\_by\_social\_context" -> HasLikedBySocialContext,

"has\_followed\_by\_social\_context" -> HasFollowedBySocialContext,

"has\_topic\_social\_context" -> HasTopicSocialContext,

"is\_followed\_topic\_tweet" -> IsFollowedTopicTweet,

"is\_recommended\_topic\_tweet" -> IsRecommendedTopicTweet,

"tweet\_age\_less\_than\_15\_seconds" -> IsTweetAgeLessThan15Seconds,

"tweet\_age\_lte\_30\_minutes" -> IsTweetAgeLessThanOrEqualTo30Minutes,

"tweet\_age\_lte\_1\_hour" -> IsTweetAgeLessThanOrEqualTo1Hour,

"tweet\_age\_lte\_6\_hours" -> IsTweetAgeLessThanOrEqualTo6Hours,

"tweet\_age\_lte\_12\_hours" -> IsTweetAgeLessThanOrEqualTo12Hours,

"tweet\_age\_gte\_24\_hours" -> IsTweetAgeGreaterThanOrEqualTo24Hours,

"has\_gte\_100\_favs" -> HasGreaterThanOrEqualTo100Favs,

"has\_gte\_1k\_favs" -> HasGreaterThanOrEqualTo1KFavs,

"has\_gte\_10k\_favs" -> HasGreaterThanOrEqualTo10KFavs,

"has\_gte\_100k\_favs" -> HasGreaterThanOrEqualTo100KFavs,

"has\_gte\_10\_retweets" -> HasGreaterThanOrEqualTo10Retweets,

"has\_gte\_100\_retweets" -> HasGreaterThanOrEqualTo100Retweets,

"has\_gte\_1k\_retweets" -> HasGreaterThanOrEqualTo1KRetweets

)

val CandidateTweetSourceIdFeatureMap: Map[Int, Binary] = Map(

CandidateTweetSourceId.RecapTweet.value -> FromRecap,

CandidateTweetSourceId.RecycledTweet.value -> FromRecycled,

CandidateTweetSourceId.RecommendedTweet.value -> FromActivity,

CandidateTweetSourceId.Simcluster.value -> FromSimcluster,

CandidateTweetSourceId.ErgTweet.value -> FromErg,

CandidateTweetSourceId.CroonTopicTweet.value -> FromCroon,

CandidateTweetSourceId.CroonTweet.value -> FromCroon,

CandidateTweetSourceId.ListTweet.value -> FromList,

CandidateTweetSourceId.RecommendedTopicTweet.value -> FromRecTopic

)

val TweetFeaturesV2: Set[Feature[\_]] = Set(

HasImage,

IsReply,

IsRetweet,

HasVisibleLink,

HasHashtag,

FromMutualFollow,

IsInNetwork

)

val ContentTweetTypeFeatures: Set[Feature[\_]] = Set(

HasImage,

HasVisibleLink,

HasHashtag,

TextOnly,

HasVisibleLink

)

val FreshnessTweetTypeFeatures: Set[Feature[\_]] = Set(

IsTweetAgeLessThan15Seconds,

IsTweetAgeLessThanOrEqualTo30Minutes,

IsTweetAgeLessThanOrEqualTo1Hour,

IsTweetAgeLessThanOrEqualTo6Hours,

IsTweetAgeLessThanOrEqualTo12Hours,

IsTweetAgeGreaterThanOrEqualTo24Hours

)

val SocialProofTweetTypeFeatures: Set[Feature[\_]] = Set(

HasLikedBySocialContext,

HasFollowedBySocialContext,

HasTopicSocialContext

)

val TopicTweetPreferenceTweetTypeFeatures: Set[Feature[\_]] = Set(

IsFollowedTopicTweet,

IsRecommendedTopicTweet

)

val TweetPopularityTweetTypeFeatures: Set[Feature[\_]] = Set(

HasGreaterThanOrEqualTo100Favs,

HasGreaterThanOrEqualTo1KFavs,

HasGreaterThanOrEqualTo10KFavs,

HasGreaterThanOrEqualTo100KFavs,

HasGreaterThanOrEqualTo10Retweets,

HasGreaterThanOrEqualTo100Retweets,

HasGreaterThanOrEqualTo1KRetweets

)

val UserGraphInteractionTweetTypeFeatures: Set[Feature[\_]] = Set(

IsInNetwork,

FromMutualFollow,

IsNotInNetwork,

IsPromoted

)

val UserContentPreferenceTweetTypeFeatures: Set[Feature[\_]] =

ContentTweetTypeFeatures ++ FreshnessTweetTypeFeatures ++ SocialProofTweetTypeFeatures ++ TopicTweetPreferenceTweetTypeFeatures ++ TweetPopularityTweetTypeFeatures ++ UserGraphInteractionTweetTypeFeatures

val AuthorContentPreferenceTweetTypeFeatures: Set[Feature[\_]] =

Set(IsInNetwork, FromMutualFollow, IsNotInNetwork) ++ ContentTweetTypeFeatures

}