package com.twitter.timelines.prediction.features.common

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

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

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

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

import com.twitter.ml.api.Feature.SparseBinary

import com.twitter.ml.api.Feature.SparseContinuous

import com.twitter.ml.api.Feature.Text

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.TypedAggregateGroup

import scala.collection.JavaConverters.\_

object TimelinesSharedFeatures extends TimelinesSharedFeatures("")

object InReplyToTweetTimelinesSharedFeatures extends TimelinesSharedFeatures("in\_reply\_to\_tweet")

/\*\*

\* Defines shared features

\*/

class TimelinesSharedFeatures(prefix: String) {

private def name(featureName: String): String = {

if (prefix.nonEmpty) {

s"$prefix.$featureName"

} else {

featureName

}

}

// meta

val EXPERIMENT\_META = new SparseBinary(

name("timelines.meta.experiment\_meta"),

Set(ExperimentId, ExperimentName).asJava)

// historically used in the "combined models" to distinguish in-network and out of network tweets.

// now the feature denotes which adapter (recap or rectweet) was used to generate the datarecords.

// and is used by the data collection pipeline to split the training data.

val INJECTION\_TYPE = new Discrete(name("timelines.meta.injection\_type"))

// Used to indicate which injection module is this

val INJECTION\_MODULE\_NAME = new Text(name("timelines.meta.injection\_module\_name"))

val LIST\_ID = new Discrete(name("timelines.meta.list\_id"))

val LIST\_IS\_PINNED = new Binary(name("timelines.meta.list\_is\_pinned"))

// internal id per each PS request. mainly to join back commomn features and candidate features later

val PREDICTION\_REQUEST\_ID = new Discrete(name("timelines.meta.prediction\_request\_id"))

// internal id per each TLM request. mainly to deduplicate re-served cached tweets in logging

val SERVED\_REQUEST\_ID = new Discrete(name("timelines.meta.served\_request\_id"))

// internal id used for join key in kafka logging, equal to servedRequestId if tweet is cached,

// else equal to predictionRequestId

val SERVED\_ID = new Discrete(name("timelines.meta.served\_id"))

val REQUEST\_JOIN\_ID = new Discrete(name("timelines.meta.request\_join\_id"))

// Internal boolean flag per tweet, whether the tweet is served from RankedTweetsCache: TQ-14050

// this feature should not be trained on, blacklisted in feature\_config: D838346

val IS\_READ\_FROM\_CACHE = new Binary(name("timelines.meta.is\_read\_from\_cache"))

// model score discounts

val PHOTO\_DISCOUNT = new Continuous(name("timelines.score\_discounts.photo"))

val VIDEO\_DISCOUNT = new Continuous(name("timelines.score\_discounts.video"))

val TWEET\_HEIGHT\_DISCOUNT = new Continuous(name("timelines.score\_discounts.tweet\_height"))

val TOXICITY\_DISCOUNT = new Continuous(name("timelines.score\_discounts.toxicity"))

// engagements

val ENGAGEMENT\_TYPE = new Discrete(name("timelines.engagement.type"))

val PREDICTED\_IS\_FAVORITED =

new Continuous(name("timelines.engagement\_predicted.is\_favorited"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_RETWEETED =

new Continuous(name("timelines.engagement\_predicted.is\_retweeted"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_QUOTED =

new Continuous(name("timelines.engagement\_predicted.is\_quoted"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_REPLIED =

new Continuous(name("timelines.engagement\_predicted.is\_replied"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_OPEN\_LINKED = new Continuous(

name("timelines.engagement\_predicted.is\_open\_linked"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_GOOD\_OPEN\_LINK = new Continuous(

name("timelines.engagement\_predicted.is\_good\_open\_link"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_PROFILE\_CLICKED = new Continuous(

name("timelines.engagement\_predicted.is\_profile\_clicked"),

Set(EngagementScore).asJava

)

val PREDICTED\_IS\_PROFILE\_CLICKED\_AND\_PROFILE\_ENGAGED = new Continuous(

name("timelines.engagement\_predicted.is\_profile\_clicked\_and\_profile\_engaged"),

Set(EngagementScore).asJava

)

val PREDICTED\_IS\_CLICKED =

new Continuous(name("timelines.engagement\_predicted.is\_clicked"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_PHOTO\_EXPANDED = new Continuous(

name("timelines.engagement\_predicted.is\_photo\_expanded"),

Set(EngagementScore).asJava

)

val PREDICTED\_IS\_FOLLOWED =

new Continuous(name("timelines.engagement\_predicted.is\_followed"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_DONT\_LIKE =

new Continuous(name("timelines.engagement\_predicted.is\_dont\_like"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_VIDEO\_PLAYBACK\_50 = new Continuous(

name("timelines.engagement\_predicted.is\_video\_playback\_50"),

Set(EngagementScore).asJava

)

val PREDICTED\_IS\_VIDEO\_QUALITY\_VIEWED = new Continuous(

name("timelines.engagement\_predicted.is\_video\_quality\_viewed"),

Set(EngagementScore).asJava

)

val PREDICTED\_IS\_GOOD\_CLICKED\_V1 = new Continuous(

name("timelines.engagement\_predicted.is\_good\_clicked\_convo\_desc\_favorited\_or\_replied"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_GOOD\_CLICKED\_V2 = new Continuous(

name("timelines.engagement\_predicted.is\_good\_clicked\_convo\_desc\_v2"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_TWEET\_DETAIL\_DWELLED\_8\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_tweet\_detail\_dwelled\_8\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_TWEET\_DETAIL\_DWELLED\_15\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_tweet\_detail\_dwelled\_15\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_TWEET\_DETAIL\_DWELLED\_25\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_tweet\_detail\_dwelled\_25\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_TWEET\_DETAIL\_DWELLED\_30\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_tweet\_detail\_dwelled\_30\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_GOOD\_CLICKED\_WITH\_DWELL\_SUM\_GTE\_60S = new Continuous(

name(

"timelines.engagement\_predicted.is\_good\_clicked\_convo\_desc\_favorited\_or\_replied\_or\_dwell\_sum\_gte\_60\_secs"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_FAVORITED\_FAV\_ENGAGED\_BY\_AUTHOR = new Continuous(

name("timelines.engagement\_predicted.is\_favorited\_fav\_engaged\_by\_author"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_REPORT\_TWEET\_CLICKED =

new Continuous(

name("timelines.engagement\_predicted.is\_report\_tweet\_clicked"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_NEGATIVE\_FEEDBACK = new Continuous(

name("timelines.engagement\_predicted.is\_negative\_feedback"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_NEGATIVE\_FEEDBACK\_V2 = new Continuous(

name("timelines.engagement\_predicted.is\_negative\_feedback\_v2"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_WEAK\_NEGATIVE\_FEEDBACK = new Continuous(

name("timelines.engagement\_predicted.is\_weak\_negative\_feedback"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_STRONG\_NEGATIVE\_FEEDBACK = new Continuous(

name("timelines.engagement\_predicted.is\_strong\_negative\_feedback"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_DWELLED\_IN\_BOUNDS\_V1 = new Continuous(

name("timelines.engagement\_predicted.is\_dwelled\_in\_bounds\_v1"),

Set(EngagementScore).asJava)

val PREDICTED\_DWELL\_NORMALIZED\_OVERALL = new Continuous(

name("timelines.engagement\_predicted.dwell\_normalized\_overall"),

Set(EngagementScore).asJava)

val PREDICTED\_DWELL\_CDF =

new Continuous(name("timelines.engagement\_predicted.dwell\_cdf"), Set(EngagementScore).asJava)

val PREDICTED\_DWELL\_CDF\_OVERALL = new Continuous(

name("timelines.engagement\_predicted.dwell\_cdf\_overall"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_DWELLED =

new Continuous(name("timelines.engagement\_predicted.is\_dwelled"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_HOME\_LATEST\_VISITED = new Continuous(

name("timelines.engagement\_predicted.is\_home\_latest\_visited"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_BOOKMARKED = new Continuous(

name("timelines.engagement\_predicted.is\_bookmarked"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_SHARED =

new Continuous(name("timelines.engagement\_predicted.is\_shared"), Set(EngagementScore).asJava)

val PREDICTED\_IS\_SHARE\_MENU\_CLICKED = new Continuous(

name("timelines.engagement\_predicted.is\_share\_menu\_clicked"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_PROFILE\_DWELLED\_20\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_profile\_dwelled\_20\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_FULLSCREEN\_VIDEO\_DWELLED\_5\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_fullscreen\_video\_dwelled\_5\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_FULLSCREEN\_VIDEO\_DWELLED\_10\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_fullscreen\_video\_dwelled\_10\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_FULLSCREEN\_VIDEO\_DWELLED\_20\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_fullscreen\_video\_dwelled\_20\_sec"),

Set(EngagementScore).asJava)

val PREDICTED\_IS\_FULLSCREEN\_VIDEO\_DWELLED\_30\_SEC = new Continuous(

name("timelines.engagement\_predicted.is\_fullscreen\_video\_dwelled\_30\_sec"),

Set(EngagementScore).asJava)

// Please use this timestamp, not the `meta.timestamp`, for the actual served timestamp.

val SERVED\_TIMESTAMP =

new Discrete("timelines.meta.timestamp.served", Set(PrivateTimestamp).asJava)

// timestamp when the engagement has occurred. do not train on these features

val TIMESTAMP\_FAVORITED =

new Discrete("timelines.meta.timestamp.engagement.favorited", Set(PublicTimestamp).asJava)

val TIMESTAMP\_RETWEETED =

new Discrete("timelines.meta.timestamp.engagement.retweeted", Set(PublicTimestamp).asJava)

val TIMESTAMP\_REPLIED =

new Discrete("timelines.meta.timestamp.engagement.replied", Set(PublicTimestamp).asJava)

val TIMESTAMP\_PROFILE\_CLICKED = new Discrete(

"timelines.meta.timestamp.engagement.profile\_clicked",

Set(PrivateTimestamp).asJava)

val TIMESTAMP\_CLICKED =

new Discrete("timelines.meta.timestamp.engagement.clicked", Set(PrivateTimestamp).asJava)

val TIMESTAMP\_PHOTO\_EXPANDED =

new Discrete("timelines.meta.timestamp.engagement.photo\_expanded", Set(PrivateTimestamp).asJava)

val TIMESTAMP\_DWELLED =

new Discrete("timelines.meta.timestamp.engagement.dwelled", Set(PrivateTimestamp).asJava)

val TIMESTAMP\_VIDEO\_PLAYBACK\_50 = new Discrete(

"timelines.meta.timestamp.engagement.video\_playback\_50",

Set(PrivateTimestamp).asJava)

// reply engaged by author

val TIMESTAMP\_REPLY\_FAVORITED\_BY\_AUTHOR = new Discrete(

"timelines.meta.timestamp.engagement.reply\_favorited\_by\_author",

Set(PublicTimestamp).asJava)

val TIMESTAMP\_REPLY\_REPLIED\_BY\_AUTHOR = new Discrete(

"timelines.meta.timestamp.engagement.reply\_replied\_by\_author",

Set(PublicTimestamp).asJava)

val TIMESTAMP\_REPLY\_RETWEETED\_BY\_AUTHOR = new Discrete(

"timelines.meta.timestamp.engagement.reply\_retweeted\_by\_author",

Set(PublicTimestamp).asJava)

// fav engaged by author

val TIMESTAMP\_FAV\_FAVORITED\_BY\_AUTHOR = new Discrete(

"timelines.meta.timestamp.engagement.fav\_favorited\_by\_author",

Set(PublicTimestamp).asJava)

val TIMESTAMP\_FAV\_REPLIED\_BY\_AUTHOR = new Discrete(

"timelines.meta.timestamp.engagement.fav\_replied\_by\_author",

Set(PublicTimestamp).asJava)

val TIMESTAMP\_FAV\_RETWEETED\_BY\_AUTHOR = new Discrete(

"timelines.meta.timestamp.engagement.fav\_retweeted\_by\_author",

Set(PublicTimestamp).asJava)

val TIMESTAMP\_FAV\_FOLLOWED\_BY\_AUTHOR = new Discrete(

"timelines.meta.timestamp.engagement.fav\_followed\_by\_author",

Set(PublicTimestamp).asJava)

// good click

val TIMESTAMP\_GOOD\_CLICK\_CONVO\_DESC\_FAVORITED = new Discrete(

"timelines.meta.timestamp.engagement.good\_click\_convo\_desc\_favorited",

Set(PrivateTimestamp).asJava)

val TIMESTAMP\_GOOD\_CLICK\_CONVO\_DESC\_REPLIIED = new Discrete(

"timelines.meta.timestamp.engagement.good\_click\_convo\_desc\_replied",

Set(PrivateTimestamp).asJava)

val TIMESTAMP\_GOOD\_CLICK\_CONVO\_DESC\_PROFILE\_CLICKED = new Discrete(

"timelines.meta.timestamp.engagement.good\_click\_convo\_desc\_profiile\_clicked",

Set(PrivateTimestamp).asJava)

val TIMESTAMP\_NEGATIVE\_FEEDBACK = new Discrete(

"timelines.meta.timestamp.engagement.negative\_feedback",

Set(PrivateTimestamp).asJava)

val TIMESTAMP\_REPORT\_TWEET\_CLICK =

new Discrete(

"timelines.meta.timestamp.engagement.report\_tweet\_click",

Set(PrivateTimestamp).asJava)

val TIMESTAMP\_IMPRESSED =

new Discrete("timelines.meta.timestamp.engagement.impressed", Set(PublicTimestamp).asJava)

val TIMESTAMP\_TWEET\_DETAIL\_DWELLED =

new Discrete(

"timelines.meta.timestamp.engagement.tweet\_detail\_dwelled",

Set(PublicTimestamp).asJava)

val TIMESTAMP\_PROFILE\_DWELLED =

new Discrete("timelines.meta.timestamp.engagement.profile\_dwelled", Set(PublicTimestamp).asJava)

val TIMESTAMP\_FULLSCREEN\_VIDEO\_DWELLED =

new Discrete(

"timelines.meta.timestamp.engagement.fullscreen\_video\_dwelled",

Set(PublicTimestamp).asJava)

val TIMESTAMP\_LINK\_DWELLED =

new Discrete("timelines.meta.timestamp.engagement.link\_dwelled", Set(PublicTimestamp).asJava)

// these are used to dup and split the negative instances during streaming processing (kafka)

val TRAINING\_FOR\_FAVORITED =

new Binary("timelines.meta.training\_data.for\_favorited", Set(EngagementId).asJava)

val TRAINING\_FOR\_RETWEETED =

new Binary("timelines.meta.training\_data.for\_retweeted", Set(EngagementId).asJava)

val TRAINING\_FOR\_REPLIED =

new Binary("timelines.meta.training\_data.for\_replied", Set(EngagementId).asJava)

val TRAINING\_FOR\_PROFILE\_CLICKED =

new Binary("timelines.meta.training\_data.for\_profile\_clicked", Set(EngagementId).asJava)

val TRAINING\_FOR\_CLICKED =

new Binary("timelines.meta.training\_data.for\_clicked", Set(EngagementId).asJava)

val TRAINING\_FOR\_PHOTO\_EXPANDED =

new Binary("timelines.meta.training\_data.for\_photo\_expanded", Set(EngagementId).asJava)

val TRAINING\_FOR\_VIDEO\_PLAYBACK\_50 =

new Binary("timelines.meta.training\_data.for\_video\_playback\_50", Set(EngagementId).asJava)

val TRAINING\_FOR\_NEGATIVE\_FEEDBACK =

new Binary("timelines.meta.training\_data.for\_negative\_feedback", Set(EngagementId).asJava)

val TRAINING\_FOR\_REPORTED =

new Binary("timelines.meta.training\_data.for\_reported", Set(EngagementId).asJava)

val TRAINING\_FOR\_DWELLED =

new Binary("timelines.meta.training\_data.for\_dwelled", Set(EngagementId).asJava)

val TRAINING\_FOR\_SHARED =

new Binary("timelines.meta.training\_data.for\_shared", Set(EngagementId).asJava)

val TRAINING\_FOR\_SHARE\_MENU\_CLICKED =

new Binary("timelines.meta.training\_data.for\_share\_menu\_clicked", Set(EngagementId).asJava)

// Warning: do not train on these features

val PREDICTED\_SCORE = new Continuous(name("timelines.score"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_FAV = new Continuous(name("timelines.score.fav"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_RETWEET =

new Continuous(name("timelines.score.retweet"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_REPLY =

new Continuous(name("timelines.score.reply"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_OPEN\_LINK =

new Continuous(name("timelines.score.open\_link"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_GOOD\_OPEN\_LINK =

new Continuous(name("timelines.score.good\_open\_link"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_PROFILE\_CLICK =

new Continuous(name("timelines.score.profile\_click"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_DETAIL\_EXPAND =

new Continuous(name("timelines.score.detail\_expand"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_PHOTO\_EXPAND =

new Continuous(name("timelines.score.photo\_expand"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_PLAYBACK\_50 =

new Continuous(name("timelines.score.playback\_50"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_VIDEO\_QUALITY\_VIEW =

new Continuous(name("timelines.score.video\_quality\_view"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_DONT\_LIKE =

new Continuous(name("timelines.score.dont\_like"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_PROFILE\_CLICKED\_AND\_PROFILE\_ENGAGED =

new Continuous(

name("timelines.score.profile\_clicked\_and\_profile\_engaged"),

Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_GOOD\_CLICKED\_V1 =

new Continuous(name("timelines.score.good\_clicked\_v1"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_GOOD\_CLICKED\_V2 =

new Continuous(name("timelines.score.good\_clicked\_v2"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_DWELL =

new Continuous(name("timelines.score.dwell"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_DWELL\_CDF =

new Continuous(name("timelines.score.dwell\_cfd"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_DWELL\_CDF\_OVERALL =

new Continuous(name("timelines.score.dwell\_cfd\_overall"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_DWELL\_NORMALIZED\_OVERALL =

new Continuous(name("timelines.score.dwell\_normalized\_overall"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_NEGATIVE\_FEEDBACK =

new Continuous(name("timelines.score.negative\_feedback"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_NEGATIVE\_FEEDBACK\_V2 =

new Continuous(name("timelines.score.negative\_feedback\_v2"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_WEAK\_NEGATIVE\_FEEDBACK =

new Continuous(name("timelines.score.weak\_negative\_feedback"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_STRONG\_NEGATIVE\_FEEDBACK =

new Continuous(name("timelines.score.strong\_negative\_feedback"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_REPORT\_TWEET\_CLICKED =

new Continuous(name("timelines.score.report\_tweet\_clicked"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_UNFOLLOW\_TOPIC =

new Continuous(name("timelines.score.unfollow\_topic"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_FOLLOW =

new Continuous(name("timelines.score.follow"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_RELEVANCE\_PROMPT\_YES\_CLICKED =

new Continuous(

name("timelines.score.relevance\_prompt\_yes\_clicked"),

Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_BOOKMARK =

new Continuous(name("timelines.score.bookmark"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_SHARE =

new Continuous(name("timelines.score.share"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_SHARE\_MENU\_CLICK =

new Continuous(name("timelines.score.share\_menu\_click"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_PROFILE\_DWELLED =

new Continuous(name("timelines.score.good\_profile\_dwelled"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_TWEET\_DETAIL\_DWELLED =

new Continuous(name("timelines.score.tweet\_detail\_dwelled"), Set(EngagementScore).asJava)

val PREDICTED\_SCORE\_FULLSCREEN\_VIDEO\_DWELL =

new Continuous(name("timelines.score.fullscreen\_video\_dwell"), Set(EngagementScore).asJava)

// hydrated in TimelinesSharedFeaturesAdapter that recap adapter calls

val ORIGINAL\_AUTHOR\_ID = new Discrete(name("entities.original\_author\_id"), Set(UserId).asJava)

val SOURCE\_AUTHOR\_ID = new Discrete(name("entities.source\_author\_id"), Set(UserId).asJava)

val SOURCE\_TWEET\_ID = new Discrete(name("entities.source\_tweet\_id"), Set(TweetId).asJava)

val TOPIC\_ID = new Discrete(name("entities.topic\_id"), Set(SemanticcoreClassification).asJava)

val INFERRED\_TOPIC\_IDS =

new SparseBinary(name("entities.inferred\_topic\_ids"), Set(SemanticcoreClassification).asJava)

val INFERRED\_TOPIC\_ID = TypedAggregateGroup.sparseFeature(INFERRED\_TOPIC\_IDS)

val WEIGHTED\_FAV\_COUNT = new Continuous(

name("timelines.earlybird.weighted\_fav\_count"),

Set(CountOfPrivateLikes, CountOfPublicLikes).asJava)

val WEIGHTED\_RETWEET\_COUNT = new Continuous(

name("timelines.earlybird.weighted\_retweet\_count"),

Set(CountOfPrivateRetweets, CountOfPublicRetweets).asJava)

val WEIGHTED\_REPLY\_COUNT = new Continuous(

name("timelines.earlybird.weighted\_reply\_count"),

Set(CountOfPrivateReplies, CountOfPublicReplies).asJava)

val WEIGHTED\_QUOTE\_COUNT = new Continuous(

name("timelines.earlybird.weighted\_quote\_count"),

Set(CountOfPrivateRetweets, CountOfPublicRetweets).asJava)

val EMBEDS\_IMPRESSION\_COUNT\_V2 = new Continuous(

name("timelines.earlybird.embeds\_impression\_count\_v2"),

Set(CountOfImpression).asJava)

val EMBEDS\_URL\_COUNT\_V2 = new Continuous(

name("timelines.earlybird.embeds\_url\_count\_v2"),

Set(CountOfPrivateTweetEntitiesAndMetadata, CountOfPublicTweetEntitiesAndMetadata).asJava)

val DECAYED\_FAVORITE\_COUNT = new Continuous(

name("timelines.earlybird.decayed\_favorite\_count"),

Set(CountOfPrivateLikes, CountOfPublicLikes).asJava)

val DECAYED\_RETWEET\_COUNT = new Continuous(

name("timelines.earlybird.decayed\_retweet\_count"),

Set(CountOfPrivateRetweets, CountOfPublicRetweets).asJava)

val DECAYED\_REPLY\_COUNT = new Continuous(

name("timelines.earlybird.decayed\_reply\_count"),

Set(CountOfPrivateReplies, CountOfPublicReplies).asJava)

val DECAYED\_QUOTE\_COUNT = new Continuous(

name("timelines.earlybird.decayed\_quote\_count"),

Set(CountOfPrivateRetweets, CountOfPublicRetweets).asJava)

val FAKE\_FAVORITE\_COUNT = new Continuous(

name("timelines.earlybird.fake\_favorite\_count"),

Set(CountOfPrivateLikes, CountOfPublicLikes).asJava)

val FAKE\_RETWEET\_COUNT = new Continuous(

name("timelines.earlybird.fake\_retweet\_count"),

Set(CountOfPrivateRetweets, CountOfPublicRetweets).asJava)

val FAKE\_REPLY\_COUNT = new Continuous(

name("timelines.earlybird.fake\_reply\_count"),

Set(CountOfPrivateReplies, CountOfPublicReplies).asJava)

val FAKE\_QUOTE\_COUNT = new Continuous(

name("timelines.earlybird.fake\_quote\_count"),

Set(CountOfPrivateRetweets, CountOfPublicRetweets).asJava)

val QUOTE\_COUNT = new Continuous(

name("timelines.earlybird.quote\_count"),

Set(CountOfPrivateRetweets, CountOfPublicRetweets).asJava)

// Safety features

val LABEL\_ABUSIVE\_FLAG =

new Binary(name("timelines.earlybird.label\_abusive\_flag"), Set(TweetSafetyLabels).asJava)

val LABEL\_ABUSIVE\_HI\_RCL\_FLAG =

new Binary(name("timelines.earlybird.label\_abusive\_hi\_rcl\_flag"), Set(TweetSafetyLabels).asJava)

val LABEL\_DUP\_CONTENT\_FLAG =

new Binary(name("timelines.earlybird.label\_dup\_content\_flag"), Set(TweetSafetyLabels).asJava)

val LABEL\_NSFW\_HI\_PRC\_FLAG =

new Binary(name("timelines.earlybird.label\_nsfw\_hi\_prc\_flag"), Set(TweetSafetyLabels).asJava)

val LABEL\_NSFW\_HI\_RCL\_FLAG =

new Binary(name("timelines.earlybird.label\_nsfw\_hi\_rcl\_flag"), Set(TweetSafetyLabels).asJava)

val LABEL\_SPAM\_FLAG =

new Binary(name("timelines.earlybird.label\_spam\_flag"), Set(TweetSafetyLabels).asJava)

val LABEL\_SPAM\_HI\_RCL\_FLAG =

new Binary(name("timelines.earlybird.label\_spam\_hi\_rcl\_flag"), Set(TweetSafetyLabels).asJava)

// Periscope features

val PERISCOPE\_EXISTS = new Binary(

name("timelines.earlybird.periscope\_exists"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val PERISCOPE\_IS\_LIVE = new Binary(

name("timelines.earlybird.periscope\_is\_live"),

Set(PrivateBroadcastMetrics, PublicBroadcastMetrics).asJava)

val PERISCOPE\_HAS\_BEEN\_FEATURED = new Binary(

name("timelines.earlybird.periscope\_has\_been\_featured"),

Set(PrivateBroadcastMetrics, PublicBroadcastMetrics).asJava)

val PERISCOPE\_IS\_CURRENTLY\_FEATURED = new Binary(

name("timelines.earlybird.periscope\_is\_currently\_featured"),

Set(PrivateBroadcastMetrics, PublicBroadcastMetrics).asJava

)

val PERISCOPE\_IS\_FROM\_QUALITY\_SOURCE = new Binary(

name("timelines.earlybird.periscope\_is\_from\_quality\_source"),

Set(PrivateBroadcastMetrics, PublicBroadcastMetrics).asJava

)

val VISIBLE\_TOKEN\_RATIO = new Continuous(name("timelines.earlybird.visible\_token\_ratio"))

val HAS\_QUOTE = new Binary(

name("timelines.earlybird.has\_quote"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val IS\_COMPOSER\_SOURCE\_CAMERA = new Binary(

name("timelines.earlybird.is\_composer\_source\_camera"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val EARLYBIRD\_SCORE = new Continuous(

name("timelines.earlybird\_score"),

Set(EngagementScore).asJava

) // separating from the rest of "timelines.earlybird." namespace

val DWELL\_TIME\_MS = new Continuous(

name("timelines.engagement.dwell\_time\_ms"),

Set(EngagementDurationAndTimestamp, ImpressionMetadata, PrivateTimestamp).asJava)

val TWEET\_DETAIL\_DWELL\_TIME\_MS = new Continuous(

name("timelines.engagement.tweet\_detail\_dwell\_time\_ms"),

Set(EngagementDurationAndTimestamp, ImpressionMetadata, PrivateTimestamp).asJava)

val PROFILE\_DWELL\_TIME\_MS = new Continuous(

name("timelines.engagement.profile\_dwell\_time\_ms"),

Set(EngagementDurationAndTimestamp, ImpressionMetadata, PrivateTimestamp).asJava)

val FULLSCREEN\_VIDEO\_DWELL\_TIME\_MS = new Continuous(

name("timelines.engagement.fullscreen\_video\_dwell\_time\_ms"),

Set(EngagementDurationAndTimestamp, ImpressionMetadata, PrivateTimestamp).asJava)

val LINK\_DWELL\_TIME\_MS = new Continuous(

name("timelines.engagement.link\_dwell\_time\_ms"),

Set(EngagementDurationAndTimestamp, ImpressionMetadata, PrivateTimestamp).asJava)

val ASPECT\_RATIO\_DEN = new Continuous(

name("tweetsource.tweet.media.aspect\_ratio\_den"),

Set(MediaFile, MediaProcessingInformation).asJava)

val ASPECT\_RATIO\_NUM = new Continuous(

name("tweetsource.tweet.media.aspect\_ratio\_num"),

Set(MediaFile, MediaProcessingInformation).asJava)

val BIT\_RATE = new Continuous(

name("tweetsource.tweet.media.bit\_rate"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HEIGHT\_2 = new Continuous(

name("tweetsource.tweet.media.height\_2"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HEIGHT\_1 = new Continuous(

name("tweetsource.tweet.media.height\_1"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HEIGHT\_3 = new Continuous(

name("tweetsource.tweet.media.height\_3"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HEIGHT\_4 = new Continuous(

name("tweetsource.tweet.media.height\_4"),

Set(MediaFile, MediaProcessingInformation).asJava)

val RESIZE\_METHOD\_1 = new Discrete(

name("tweetsource.tweet.media.resize\_method\_1"),

Set(MediaFile, MediaProcessingInformation).asJava)

val RESIZE\_METHOD\_2 = new Discrete(

name("tweetsource.tweet.media.resize\_method\_2"),

Set(MediaFile, MediaProcessingInformation).asJava)

val RESIZE\_METHOD\_3 = new Discrete(

name("tweetsource.tweet.media.resize\_method\_3"),

Set(MediaFile, MediaProcessingInformation).asJava)

val RESIZE\_METHOD\_4 = new Discrete(

name("tweetsource.tweet.media.resize\_method\_4"),

Set(MediaFile, MediaProcessingInformation).asJava)

val VIDEO\_DURATION = new Continuous(

name("tweetsource.tweet.media.video\_duration"),

Set(MediaFile, MediaProcessingInformation).asJava)

val WIDTH\_1 = new Continuous(

name("tweetsource.tweet.media.width\_1"),

Set(MediaFile, MediaProcessingInformation).asJava)

val WIDTH\_2 = new Continuous(

name("tweetsource.tweet.media.width\_2"),

Set(MediaFile, MediaProcessingInformation).asJava)

val WIDTH\_3 = new Continuous(

name("tweetsource.tweet.media.width\_3"),

Set(MediaFile, MediaProcessingInformation).asJava)

val WIDTH\_4 = new Continuous(

name("tweetsource.tweet.media.width\_4"),

Set(MediaFile, MediaProcessingInformation).asJava)

val NUM\_MEDIA\_TAGS = new Continuous(

name("tweetsource.tweet.media.num\_tags"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val MEDIA\_TAG\_SCREEN\_NAMES = new SparseBinary(

name("tweetsource.tweet.media.tag\_screen\_names"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val STICKER\_IDS = new SparseBinary(

name("tweetsource.tweet.media.sticker\_ids"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val NUM\_COLOR\_PALLETTE\_ITEMS = new Continuous(

name("tweetsource.v2.tweet.media.num\_color\_pallette\_items"),

Set(MediaFile, MediaProcessingInformation).asJava)

val COLOR\_1\_RED = new Continuous(

name("tweetsource.v2.tweet.media.color\_1\_red"),

Set(MediaFile, MediaProcessingInformation).asJava)

val COLOR\_1\_BLUE = new Continuous(

name("tweetsource.v2.tweet.media.color\_1\_blue"),

Set(MediaFile, MediaProcessingInformation).asJava)

val COLOR\_1\_GREEN = new Continuous(

name("tweetsource.v2.tweet.media.color\_1\_green"),

Set(MediaFile, MediaProcessingInformation).asJava)

val COLOR\_1\_PERCENTAGE = new Continuous(

name("tweetsource.v2.tweet.media.color\_1\_percentage"),

Set(MediaFile, MediaProcessingInformation).asJava)

val MEDIA\_PROVIDERS = new SparseBinary(

name("tweetsource.v2.tweet.media.providers"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val IS\_360 = new Binary(

name("tweetsource.v2.tweet.media.is\_360"),

Set(MediaFile, MediaProcessingInformation).asJava)

val VIEW\_COUNT =

new Continuous(name("tweetsource.v2.tweet.media.view\_count"), Set(MediaContentMetrics).asJava)

val IS\_MANAGED = new Binary(

name("tweetsource.v2.tweet.media.is\_managed"),

Set(MediaFile, MediaProcessingInformation).asJava)

val IS\_MONETIZABLE = new Binary(

name("tweetsource.v2.tweet.media.is\_monetizable"),

Set(MediaFile, MediaProcessingInformation).asJava)

val IS\_EMBEDDABLE = new Binary(

name("tweetsource.v2.tweet.media.is\_embeddable"),

Set(MediaFile, MediaProcessingInformation).asJava)

val CLASSIFICATION\_LABELS = new SparseContinuous(

name("tweetsource.v2.tweet.media.classification\_labels"),

Set(MediaFile, MediaProcessingInformation).asJava)

val NUM\_STICKERS = new Continuous(

name("tweetsource.v2.tweet.media.num\_stickers"),

Set(PublicTweetEntitiesAndMetadata, PrivateTweetEntitiesAndMetadata).asJava)

val NUM\_FACES = new Continuous(

name("tweetsource.v2.tweet.media.num\_faces"),

Set(MediaFile, MediaProcessingInformation).asJava)

val FACE\_AREAS = new Continuous(

name("tweetsource.v2.tweet.media.face\_areas"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HAS\_SELECTED\_PREVIEW\_IMAGE = new Binary(

name("tweetsource.v2.tweet.media.has\_selected\_preview\_image"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HAS\_TITLE = new Binary(

name("tweetsource.v2.tweet.media.has\_title"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HAS\_DESCRIPTION = new Binary(

name("tweetsource.v2.tweet.media.has\_description"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HAS\_VISIT\_SITE\_CALL\_TO\_ACTION = new Binary(

name("tweetsource.v2.tweet.media.has\_visit\_site\_call\_to\_action"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HAS\_APP\_INSTALL\_CALL\_TO\_ACTION = new Binary(

name("tweetsource.v2.tweet.media.has\_app\_install\_call\_to\_action"),

Set(MediaFile, MediaProcessingInformation).asJava)

val HAS\_WATCH\_NOW\_CALL\_TO\_ACTION = new Binary(

name("tweetsource.v2.tweet.media.has\_watch\_now\_call\_to\_action"),

Set(MediaFile, MediaProcessingInformation).asJava)

val NUM\_CAPS =

new Continuous(name("tweetsource.tweet.text.num\_caps"), Set(PublicTweets, PrivateTweets).asJava)

val TWEET\_LENGTH =

new Continuous(name("tweetsource.tweet.text.length"), Set(PublicTweets, PrivateTweets).asJava)

val TWEET\_LENGTH\_TYPE = new Discrete(

name("tweetsource.tweet.text.length\_type"),

Set(PublicTweets, PrivateTweets).asJava)

val NUM\_WHITESPACES = new Continuous(

name("tweetsource.tweet.text.num\_whitespaces"),

Set(PublicTweets, PrivateTweets).asJava)

val HAS\_QUESTION =

new Binary(name("tweetsource.tweet.text.has\_question"), Set(PublicTweets, PrivateTweets).asJava)

val NUM\_NEWLINES = new Continuous(

name("tweetsource.tweet.text.num\_newlines"),

Set(PublicTweets, PrivateTweets).asJava)

val EMOJI\_TOKENS = new SparseBinary(

name("tweetsource.v3.tweet.text.emoji\_tokens"),

Set(PublicTweets, PrivateTweets).asJava)

val EMOTICON\_TOKENS = new SparseBinary(

name("tweetsource.v3.tweet.text.emoticon\_tokens"),

Set(PublicTweets, PrivateTweets).asJava)

val NUM\_EMOJIS = new Continuous(

name("tweetsource.v3.tweet.text.num\_emojis"),

Set(PublicTweets, PrivateTweets).asJava)

val NUM\_EMOTICONS = new Continuous(

name("tweetsource.v3.tweet.text.num\_emoticons"),

Set(PublicTweets, PrivateTweets).asJava)

val POS\_UNIGRAMS = new SparseBinary(

name("tweetsource.v3.tweet.text.pos\_unigrams"),

Set(PublicTweets, PrivateTweets).asJava)

val POS\_BIGRAMS = new SparseBinary(

name("tweetsource.v3.tweet.text.pos\_bigrams"),

Set(PublicTweets, PrivateTweets).asJava)

val TEXT\_TOKENS = new SparseBinary(

name("tweetsource.v4.tweet.text.tokens"),

Set(PublicTweets, PrivateTweets).asJava)

// Health features model scores (see go/toxicity, go/pblock, go/pspammytweet)

val PBLOCK\_SCORE =

new Continuous(name("timelines.earlybird.pblock\_score"), Set(TweetSafetyScores).asJava)

val TOXICITY\_SCORE =

new Continuous(name("timelines.earlybird.toxicity\_score"), Set(TweetSafetyScores).asJava)

val EXPERIMENTAL\_HEALTH\_MODEL\_SCORE\_1 =

new Continuous(

name("timelines.earlybird.experimental\_health\_model\_score\_1"),

Set(TweetSafetyScores).asJava)

val EXPERIMENTAL\_HEALTH\_MODEL\_SCORE\_2 =

new Continuous(

name("timelines.earlybird.experimental\_health\_model\_score\_2"),

Set(TweetSafetyScores).asJava)

val EXPERIMENTAL\_HEALTH\_MODEL\_SCORE\_3 =

new Continuous(

name("timelines.earlybird.experimental\_health\_model\_score\_3"),

Set(TweetSafetyScores).asJava)

val EXPERIMENTAL\_HEALTH\_MODEL\_SCORE\_4 =

new Continuous(

name("timelines.earlybird.experimental\_health\_model\_score\_4"),

Set(TweetSafetyScores).asJava)

val PSPAMMY\_TWEET\_SCORE =

new Continuous(name("timelines.earlybird.pspammy\_tweet\_score"), Set(TweetSafetyScores).asJava)

val PREPORTED\_TWEET\_SCORE =

new Continuous(name("timelines.earlybird.preported\_tweet\_score"), Set(TweetSafetyScores).asJava)

// where record was displayed e.g. recap vs ranked timeline vs recycled

// (do NOT use for training in prediction, since this is set post-scoring)

// This differs from TimelinesSharedFeatures.INJECTION\_TYPE, which is only

// set to Recap or Rectweet, and is available pre-scoring.

// This also differs from TimeFeatures.IS\_TWEET\_RECYCLED, which is set

// pre-scoring and indicates if a tweet is being considered for recycling.

// In contrast, DISPLAY\_SUGGEST\_TYPE == RecycledTweet means the tweet

// was actually served in a recycled tweet module. The two should currently

// have the same value, but need not in future, so please only use

// IS\_TWEET\_RECYCLED/CANDIDATE\_TWEET\_SOURCE\_ID for training models and

// only use DISPLAY\_SUGGEST\_TYPE for offline analysis of tweets actually

// served in recycled modules.

val DISPLAY\_SUGGEST\_TYPE = new Discrete(name("recap.display.suggest\_type"))

// Candidate tweet source id - related to DISPLAY\_SUGGEST\_TYPE above, but this is a

// property of the candidate rather than display location so is safe to use

// in model training, unlike DISPLAY\_SUGGEST\_TYPE.

val CANDIDATE\_TWEET\_SOURCE\_ID =

new Discrete(name("timelines.meta.candidate\_tweet\_source\_id"), Set(TweetId).asJava)

// Was at least 50% of this tweet in the user's viewport for at least 500 ms,

// OR did the user engage with the tweet publicly or privately

val IS\_LINGER\_IMPRESSION =

new Binary(name("timelines.engagement.is\_linger\_impression"), Set(EngagementsPrivate).asJava)

// Features to create rollups

val LANGUAGE\_GROUP = new Discrete(name("timelines.tweet.text.language\_group"))

// The final position index of the tweet being trained on in the timeline

// served from TLM (could still change later in TLS-API), as recorded by

// PositionIndexLoggingEnvelopeTransform.

val FINAL\_POSITION\_INDEX = new Discrete(name("timelines.display.final\_position\_index"))

// The traceId of the timeline request, can be used to group tweets in the same response.

val TRACE\_ID = new Discrete(name("timelines.display.trace\_id"), Set(TfeTransactionId).asJava)

// Whether this tweet was randomly injected into the timeline or not, for exploration purposes

val IS\_RANDOM\_TWEET = new Binary(name("timelines.display.is\_random\_tweet"))

// Whether this tweet was reordered with softmax ranking for explore/exploit, and needs to

// be excluded from exploit only holdback

val IS\_SOFTMAX\_RANKING\_TWEET = new Binary(name("timelines.display.is\_softmax\_ranking\_tweet"))

// Whether the user viewing the tweet has disabled ranked timeline.

val IS\_RANKED\_TIMELINE\_DISABLER = new Binary(

name("timelines.user\_features.is\_ranked\_timeline\_disabler"),

Set(AnnotationValue, GeneralSettings).asJava)

// Whether the user viewing the tweet was one of those released from DDG 4205 control

// as part of http://go/shrink-4205 process to shrink the quality features holdback.

val IS\_USER\_RELEASED\_FROM\_QUALITY\_HOLDBACK = new Binary(

name("timelines.user\_features.is\_released\_from\_quality\_holdback"),

Set(ExperimentId, ExperimentName).asJava)

val INITIAL\_PREDICTION\_FAV =

new Continuous(name("timelines.initial\_prediction.fav"), Set(EngagementScore).asJava)

val INITIAL\_PREDICTION\_RETWEET =

new Continuous(name("timelines.initial\_prediction.retweet"), Set(EngagementScore).asJava)

val INITIAL\_PREDICTION\_REPLY =

new Continuous(name("timelines.initial\_prediction.reply"), Set(EngagementScore).asJava)

val INITIAL\_PREDICTION\_OPEN\_LINK =

new Continuous(name("timelines.initial\_prediction.open\_link"), Set(EngagementScore).asJava)

val INITIAL\_PREDICTION\_PROFILE\_CLICK =

new Continuous(name("timelines.initial\_prediction.profile\_click"), Set(EngagementScore).asJava)

val INITIAL\_PREDICTION\_VIDEO\_PLAYBACK\_50 = new Continuous(

name("timelines.initial\_prediction.video\_playback\_50"),

Set(EngagementScore).asJava)

val INITIAL\_PREDICTION\_DETAIL\_EXPAND =

new Continuous(name("timelines.initial\_prediction.detail\_expand"), Set(EngagementScore).asJava)

val INITIAL\_PREDICTION\_PHOTO\_EXPAND =

new Continuous(name("timelines.initial\_prediction.photo\_expand"), Set(EngagementScore).asJava)

val VIEWER\_FOLLOWS\_ORIGINAL\_AUTHOR =

new Binary(name("timelines.viewer\_follows\_original\_author"), Set(Follow).asJava)

val IS\_TOP\_ONE = new Binary(name("timelines.position.is\_top\_one"))

val IS\_TOP\_FIVE =

new Binary(name(featureName = "timelines.position.is\_top\_five"))

val IS\_TOP\_TEN =

new Binary(name(featureName = "timelines.position.is\_top\_ten"))

val LOG\_POSITION =

new Continuous(name(featureName = "timelines.position.log\_10"))

}