```
//Projection, excluded because Neo4J does not allow string in projection
CALL gds.graph.project(
  'author-coauthorship-graph',
  ['Author'],
  {
    CO_AUTHORED: {
      orientation: 'UNDIRECTED',
      aggregation: 'SINGLE'
    }
  }
);


// Pipeline created
CALL gds.beta.pipeline.linkPrediction.create('author-lp');


// degree and fastRP properties added
CALL gds.beta.pipeline.linkPrediction.addNodeProperty('author-lp', 'degree', {
  mutateProperty: 'degree'
});

CALL gds.beta.pipeline.linkPrediction.addNodeProperty('author-lp', 'fastRP', {
  mutateProperty: 'embedding',
  embeddingDimension: 64
});


// hadmard chosen for feature extraction
CALL gds.beta.pipeline.linkPrediction.addFeature('author-lp', 'hadamard', {
  nodeProperties: ['embedding']
});


// Train & Test configuration
CALL gds.beta.pipeline.linkPrediction.configureSplit('author-lp', {
  trainFraction: 0.1,
  testFraction: 0.1,
  negativeSamplingRatio: 1.0
});


// Random forest chosen as the prediction model
CALL gds.beta.pipeline.linkPrediction.addRandomForest('author-lp', {
```

```
  numberOfDecisionTrees: 10
});


// Model Training
CALL gds.beta.pipeline.linkPrediction.train('author-coauthorship-graph', {
  pipeline: 'author-lp',
  modelName: 'author-lp-model',
  targetRelationshipType: 'CO_AUTHORED'
});



// Link Prediction
CALL gds.beta.pipeline.linkPrediction.predict.stream('author-coauthorship-graph', {
  modelName: 'author-lp-model',
  writeRelationshipType: 'PREDICTED_CO_AUTHOR',
  topN: 2
})
YIELD node1, node2, probability
WITH gds.util.asNode(node1) AS author1, gds.util.asNode(node2) AS author2, probability
WHERE NOT (author1)-[:CO_AUTHORED]-(author2)
RETURN author1.name AS Author1, author2.name AS Author2, probability
ORDER BY probability DESC
LIMIT 20;
```