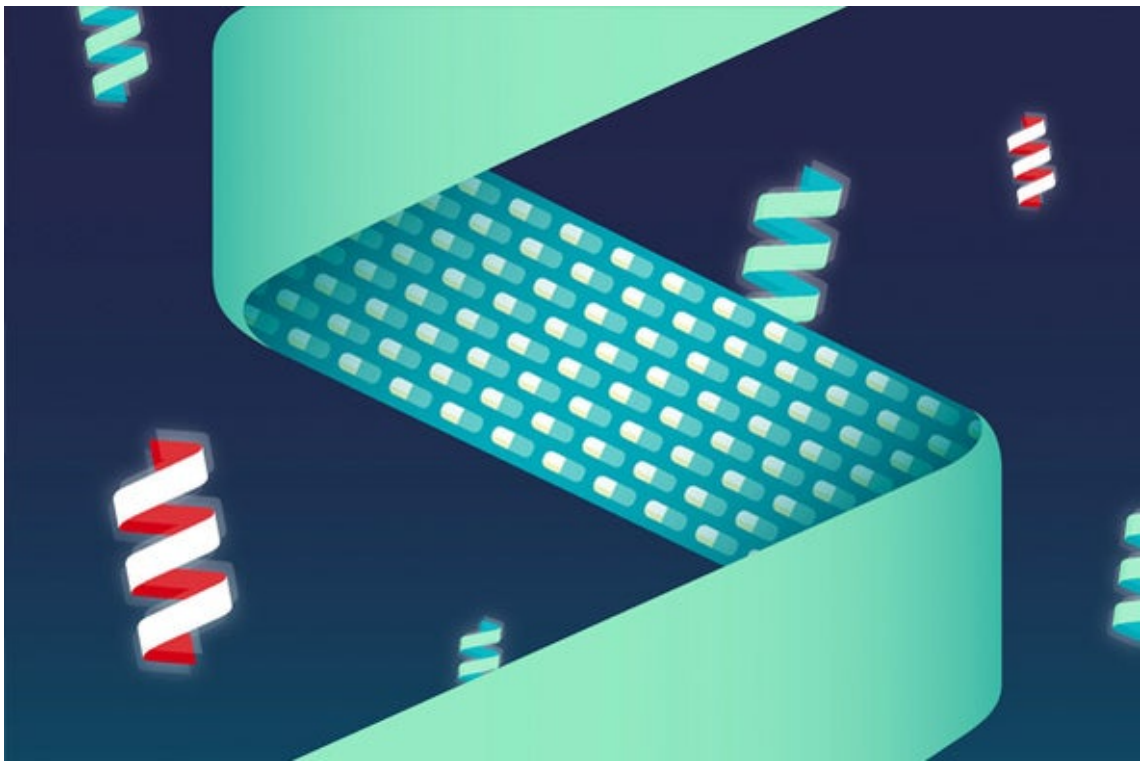
UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Proteins-Recommendation System for Drugs



Simone Contini

Indice

1 Introduzione

In questo progetto viene implementato un sistema di raccomandazione che, considerata la rete Protein-Protein Interaction (PPI) dal database *Intact* (https://www.ebi.ac.uk/intact/search) e considerate le associazioni tra drugs e targets dal database *DrugBank* (https://www.drugbank.ca/), in funzione di come sono collegate le proteine sulla rete PPI, data una drug suggerisce nuovi target.

Viene, infine, effettuata un'analisi accurata dei risultati attraverso l'implementazione di grafi interattivi che raffigurano in maniera intuitiva le interazioni tra le proteine predette e le proteine che hanno una relazione diretta con la drug scelta per il sistema di raccomandazione. Dai risultati mostrati su grafi si evince come le proteine predette dal sistema per una determinata drug possano essere effettivamente dei target appropriati per la stessa.

2 Progettazione

L'intero lavoro è stato implementato nel linguaggio di programmazione *Python*, utilizzando *Atom* e *Jupyter Notebook* come IDE.

Le fasi di progettazione sono le seguenti:

- Importazione delle librerie necessarie e configurazione per l'inizializzazione di Spark;
- Creazione e pulitura dei DataFrame della rete PPI e di DrugBank;
- Creazione del DataFrame da utilizzare per il sistema di raccomandazione;
- Conversione dei valori di tipo String in indici, necessari per il sistema di raccomandazione;
- Creazione del modello ALS ed addestramento dei dati;
- Esecuzione del sistema di raccomandazione;
- Conversione degli indici nelle stringhe di origine;
- Visualizzazione del sistema di raccomandazione;
- Generazione e visualizzazione dei grafi.

2.1 Librerie e Settaggi

Per il seguente lavoro, sono state sfruttate le seguenti librerie:

- PySpark;
- Pandas;
- Numpy;
- NetworkX;
- PyVis.

```
import networkx as nx
#import matplotlib.pyplot as plt
import os
import pandas
import numpy as np

from pyspark.sql import SparkSession
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.ml.tuning import TrainValidationSplit, ParamGridBuilder
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.ml.feature import StringIndexer, IndexToString
from pyspark.ml import Pipeline
from pyspark.sql.functions import explode
from pyspark.sql.functions import arrays_overlap, array
from pyspark.sql.functions import col,array_contains
from pyspark.sql.functions import expr
from pyvis.network import Network

spark = SparkSession.builder.appName('Recommendation_system').getOrCreate()
```

2.2 Creazione e pulizia dei DataFrame

Due sono i dataset utilizzati per questo lavoro: per la rete Protein-Protein Interaction dell'uomo è stato considerato il database *Intact* (https://www.ebi.ac.uk/intact/search), contenente 15 feature (ID(s) interactor A, ID(s) interactor B, Alt. ID(s) interactor A, Alt. ID(s) interactor B, Alias(es) interactor A, Alias(es) interactor B, Interaction detection method(s), Publication 1st author(s), Publication Identifier(s), Taxid interactor A, Taxid interactor B, Interaction type(s), Source database(s), Interaction identifier(s), Confidence values(s)) e 1054920 record; infine, per le associazioni dirette tra drug e target è stato utilizzato il database *DrugBank* (https://www.drugbank.ca/), contenente 5 feature (DrugBank ID, Name, Type, UniProt ID, UniProt Name) e 20941 record.

Entrambi i file sono stati scaricati in formato CSV e sono stati caricati in forma di DataFrame (utilizzando apposite funzioni della libreria PySpark) per poter essere elaborati.

```
ppiDF = spark.read.csv("DATA/species_13.csv", header=True, inferSchema=True, sep='\t')
ppiDF.show()
```

```
+------------------+-----------------+----------------------+----------------------+---------------------+---
------------------+-------------------------------+------------------------+------------------------+----------
----------+--------------------+--------------------+--------------------+------------------------+-------------
-----+
|#ID(s) interactor A|ID(s) interactor B|Alt. ID(s) interactor A|Alt. ID(s) interactor B|Alias(es) interactor A|Ali
as(es) interactor B|Interaction detection method(s)|Publication 1st author(s)|Publication Identifier(s)|  Taxid in
teractor A|  Taxid interactor B| Interaction type(s)|  Source database(s)|Interaction identifier(s)|Confidence val
ue(s)|
+------------------+-----------------+----------------------+----------------------+---------------------+---
------------------+-------------------------------+------------------------+------------------------+----------
----------+--------------------+--------------------+--------------------+------------------------+-------------
-----+
|   uniprotkb:P38764|  uniprotkb:P40016|   intact:EBI-15913|...|   intact:EBI-15927|...|  psi-mi:rpn1_yeast...|  p
si-mi:rpn3_yeast...|           psi-mi:"MI:0676"(...|     Krogan et al. (2006)|     pubmed:16554755|i...|taxid:5592
92(yeas...|taxid:559292(yeas...|psi-mi:"MI:0915"(...|psi-mi:"MI:0471"(...|     intact:EBI-694186...|intact-miscor
e:0.76|
|   uniprotkb:Q01939|  uniprotkb:P40016|   intact:EBI-13914|...|   intact:EBI-15927|...|  psi-mi:prs8_yeast...|  p
si-mi:rpn3_yeast...|           psi-mi:"MI:0676"(...|     Krogan et al. (2006)|     pubmed:16554755|i...|taxid:5592
92(yeas...|taxid:559292(yeas...|psi-mi:"MI:0915"(...|psi-mi:"MI:0471"(...|     intact:EBI-694187...|intact-miscor
e:0.40|
|   uniprotkb:P33299|  uniprotkb:P40016|   intact:EBI-13910|...|   intact:EBI-15927|...|  psi-mi:prs7_yeast...|  p
si-mi:rpn3_yeast...|           psi-mi:"MI:0676"(...|     Krogan et al. (2006)|     pubmed:16554755|i...|taxid:5592
92(yeas...|taxid:559292(yeas...|psi-mi:"MI:0915"(...|psi-mi:"MI:0471"(...|     intact:EBI-694190...|intact-miscor
e:0.69|
|   uniprotkb:Q06103|  uniprotkb:P40016|   intact:EBI-15940|...|   intact:EBI-15927|...|  psi-mi:rpn7_yeast...|  p
si-mi:rpn3_yeast...|           psi-mi:"MI:0676"(...|     Krogan et al. (2006)|     pubmed:16554755|i...|taxid:5592
92(yeas...|taxid:559292(yeas...|psi-mi:"MI:0915"(...|psi-mi:"MI:0471"(...|     intact:EBI-694189...|intact-miscor
e:0.81|
```

```
drugBankDF = spark.read.csv("DATA/uniprot links.csv", header=True, inferSchema=True)
drugBankDF.show()
```

```
+----------+------------------+----------+----------+--------------------+
|DrugBank ID|              Name|      Type|UniProt ID|       UniProt Name|
+----------+------------------+----------+----------+--------------------+
|    DB00001|          Lepirudin|BiotechDrug|    P00734|         Prothrombin|
|    DB00002|          Cetuximab|BiotechDrug|    P00533|Epidermal growth ...|
|    DB00002|          Cetuximab|BiotechDrug|    O75015|Low affinity immu...|
|    DB00002|          Cetuximab|BiotechDrug|    P00736|Complement C1r su...|
|    DB00002|          Cetuximab|BiotechDrug|    P02745|Complement C1q su...|
|    DB00002|          Cetuximab|BiotechDrug|    P02746|Complement C1q su...|
|    DB00002|          Cetuximab|BiotechDrug|    P02747|Complement C1q su...|
|    DB00002|          Cetuximab|BiotechDrug|    P08637|Low affinity immu...|
|    DB00002|          Cetuximab|BiotechDrug|    P09871|Complement C1s su...|
|    DB00002|          Cetuximab|BiotechDrug|    P12314|High affinity imm...|
|    DB00002|          Cetuximab|BiotechDrug|    P12318|Low affinity immu...|
|    DB00002|          Cetuximab|BiotechDrug|    P31994|Low affinity immu...|
|    DB00002|          Cetuximab|BiotechDrug|    P31995|Low affinity immu...|
|    DB00004|Denileukin diftitox|BiotechDrug|    P01589|Interleukin-2 rec...|
|    DB00004|Denileukin diftitox|BiotechDrug|    P14784|Interleukin-2 rec...|
|    DB00004|Denileukin diftitox|BiotechDrug|    P31785|Cytokine receptor...|
|    DB00005|         Etanercept|BiotechDrug|    P01375|Tumor necrosis fa...|
|    DB00005|         Etanercept|BiotechDrug|    P20333|Tumor necrosis fa...|
|    DB00005|         Etanercept|BiotechDrug|    P12314|High affinity imm...|
|    DB00005|         Etanercept|BiotechDrug|    P08637|Low affinity immu...|
+----------+------------------+----------+----------+--------------------+
only showing top 20 rows
```

Dal momento che molte feature, in entrambi i DataFrame, non sono utili ai fini del progetto, sono state selezionate soltanto quelle necessarie:

```
drugBankDF = drugBankDF.select(drugBankDF["DrugBank ID"], drugBankDF["UniProt ID"])
drugBankDF.show()
```

```
+-----------+----------+
|DrugBank ID|UniProt ID|
+-----------+----------+
|    DB00001|    P00734|
|    DB00002|    P00533|
|    DB00002|    O75015|
|    DB00002|    P00736|
|    DB00002|    P02745|
|    DB00002|    P02746|
|    DB00002|    P02747|
|    DB00002|    P08637|
|    DB00002|    P09871|
|    DB00002|    P12314|
|    DB00002|    P12318|
|    DB00002|    P31994|
|    DB00002|    P31995|
|    DB00004|    P01589|
|    DB00004|    P14784|
|    DB00004|    P31785|
|    DB00005|    P01375|
|    DB00005|    P20333|
|    DB00005|    P12314|
|    DB00005|    P08637|
+-----------+----------+
only showing top 20 rows
```

```
ppiDF = ppiDF.select(ppiDF['#ID(s) interactor A'], ppiDF['ID(s) interactor B'], ppiDF['Confidence value(s)'])
ppiDF.show()
```

```
+------------------+------------------+------------------+
|#ID(s) interactor A|ID(s) interactor B|Confidence value(s)|
+------------------+------------------+------------------+
|   uniprotkb:P38764|  uniprotkb:P40016|intact-miscore:0.76|
|   uniprotkb:Q01939|  uniprotkb:P40016|intact-miscore:0.40|
|   uniprotkb:P33299|  uniprotkb:P40016|intact-miscore:0.69|
|   uniprotkb:Q06103|  uniprotkb:P40016|intact-miscore:0.81|
|   uniprotkb:P38764|  uniprotkb:P40016|intact-miscore:0.76|
|   uniprotkb:P40016|  uniprotkb:P38764|intact-miscore:0.76|
|   uniprotkb:P53549|  uniprotkb:P40016|intact-miscore:0.55|
|   uniprotkb:P40016|  uniprotkb:Q08723|intact-miscore:0.70|
|   uniprotkb:Q08723|  uniprotkb:P40016|intact-miscore:0.70|
|   uniprotkb:P40016|  uniprotkb:P38886|intact-miscore:0.76|
|   uniprotkb:P40016|  uniprotkb:P43588|intact-miscore:0.76|
|   uniprotkb:P40016|  uniprotkb:P38764|intact-miscore:0.76|
|   uniprotkb:P40016|  uniprotkb:Q06103|intact-miscore:0.81|
|   uniprotkb:P53008|  uniprotkb:P40016|intact-miscore:0.40|
|   uniprotkb:P32565|  uniprotkb:P40016|intact-miscore:0.40|
|   uniprotkb:P40016|  uniprotkb:P32565|intact-miscore:0.40|
|   uniprotkb:P38764|  uniprotkb:P40016|intact-miscore:0.76|
|   uniprotkb:P53549|  uniprotkb:P40016|intact-miscore:0.55|
|   uniprotkb:P40016|  uniprotkb:P38764|intact-miscore:0.76|
|   uniprotkb:Q08723|  uniprotkb:P40016|intact-miscore:0.70|
+------------------+------------------+------------------+
only showing top 20 rows
```

Come si può ben vedere in ppiDF, i record di ciascuna feature presentano valori "sporchi" (presenza delle parole uniprotkb:, chebi:, ensembl:, ensemblgenomes:, intact:, refseq:, intact-miscore:). Sia per una questione di pulizia visuale, sia, sopratutto, perchè i record di Confidence values(s) serviranno successivamente come valori di tipo Float, viene effettuata l'eliminazione di tali parole in ciascun record di ciascuna feature del DataFrame. Inoltre, vengono ridenominati i nomi delle colonne:

```
ppiDF = ppiDF.withColumn('#ID(s) interactor A', regexp_replace('#ID(s) interactor A', 'uniprotkb:', ''))
ppiDF = ppiDF.withColumn('#ID(s) interactor A', regexp_replace('#ID(s) interactor A', 'chebi:', ''))
ppiDF = ppiDF.withColumn('#ID(s) interactor A', regexp_replace('#ID(s) interactor A', 'ensembl:', ''))
ppiDF = ppiDF.withColumn('#ID(s) interactor A', regexp_replace('#ID(s) interactor A', 'ensemblgenomes:', ''))
ppiDF = ppiDF.withColumn('#ID(s) interactor A', regexp_replace('#ID(s) interactor A', 'intact:', ''))
ppiDF = ppiDF.withColumn('#ID(s) interactor A', regexp_replace('#ID(s) interactor A', 'refseq:', ''))

ppiDF = ppiDF.withColumn('ID(s) interactor B', regexp_replace('ID(s) interactor B', 'uniprotkb:', ''))
ppiDF = ppiDF.withColumn('ID(s) interactor B', regexp_replace('ID(s) interactor B', 'chebi:', ''))
ppiDF = ppiDF.withColumn('ID(s) interactor B', regexp_replace('ID(s) interactor B', 'ensembl:', ''))
ppiDF = ppiDF.withColumn('ID(s) interactor B', regexp_replace('ID(s) interactor B', 'ensemblgenomes:', ''))
ppiDF = ppiDF.withColumn('ID(s) interactor B', regexp_replace('ID(s) interactor B', 'intact:', ''))
ppiDF = ppiDF.withColumn('ID(s) interactor B', regexp_replace('ID(s) interactor B', 'refseq:', ''))

ppiDF = ppiDF.withColumn('Confidence value(s)', regexp_replace('Confidence value(s)', 'intact-miscore:', ''))
```

```
ppiDF = ppiDF.withColumnRenamed("#ID(s) interactor A", "Interactor_A")
ppiDF = ppiDF.withColumnRenamed("ID(s) interactor B", "Interactor_B")
ppiDF = ppiDF.withColumnRenamed("Confidence value(s)", "Confidence")

ppiDF.show()
```

```
+------------+------------+----------+
|Interactor_A|Interactor_B|Confidence|
+------------+------------+----------+
|      P38764|      P40016|      0.76|
|      Q01939|      P40016|      0.40|
|      P33299|      P40016|      0.69|
|      Q06103|      P40016|      0.81|
|      P38764|      P40016|      0.76|
|      P40016|      P38764|      0.76|
|      P53549|      P40016|      0.55|
|      P40016|      Q08723|      0.70|
|      Q08723|      P40016|      0.70|
|      P40016|      P38886|      0.76|
|      P40016|      P43588|      0.76|
|      P40016|      P38764|      0.76|
|      P40016|      Q06103|      0.81|
|      P53008|      P40016|      0.40|
|      P32565|      P40016|      0.40|
|      P40016|      P32565|      0.40|
|      P38764|      P40016|      0.76|
|      P53549|      P40016|      0.55|
|      P40016|      P38764|      0.76|
|      Q08723|      P40016|      0.70|
+------------+------------+----------+
only showing top 20 rows
```

Allo stesso modo, vengono ridenominati i nomi delle feature in drugBankDF:

```
drugBankDF = drugBankDF.withColumnRenamed("DrugBank ID", "ID_DrugBank")
drugBankDF = drugBankDF.withColumnRenamed("UniProt ID", "ID_UniProt")

drugBankDF.show()
```

```
+-----------+----------+
|ID_DrugBank|ID_UniProt|
+-----------+----------+
|    DB00001|    P00734|
|    DB00002|    P00533|
|    DB00002|    O75015|
|    DB00002|    P00736|
|    DB00002|    P02745|
|    DB00002|    P02746|
|    DB00002|    P02747|
|    DB00002|    P08637|
|    DB00002|    P09871|
|    DB00002|    P12314|
|    DB00002|    P12318|
|    DB00002|    P31994|
|    DB00002|    P31995|
|    DB00004|    P01589|
|    DB00004|    P14784|
|    DB00004|    P31785|
|    DB00005|    P01375|
|    DB00005|    P20333|
|    DB00005|    P12314|
|    DB00005|    P08637|
+-----------+----------+
only showing top 20 rows
```

Il passo successivo consiste nel convertire i record di Confidence in ppiDF (valori di tipo String) in valori di tipo Float. Inoltre, vengono eliminati tutti i record che presentano valori di Confidence pari a *Null*:

```
ppiDF = ppiDF.na.drop(subset=["Confidence"])
ppiDF.filter(ppiDF["Confidence"].isNull()).show()
```

```
+------------+------------+----------+
|Interactor_A|Interactor_B|Confidence|
+------------+------------+----------+
+------------+------------+----------+
```

```
ppiDF = ppiDF.withColumn("Confidence", ppiDF["Confidence"].cast(FloatType()))
ppiDF.printSchema()
```

```
root
 |-- Interactor_A: string (nullable = true)
 |-- Interactor_B: string (nullable = true)
 |-- Confidence: float (nullable = true)
```

I DataFrame ppiDF e drugBankDF opportunamente puliti sono, dunque, i seguenti:

```
ppiDF.show()
```

```
+------------+------------+----------+
|Interactor_A|Interactor_B|Confidence|
+------------+------------+----------+
|      P38764|      P40016|      0.76|
|      Q01939|      P40016|       0.4|
|      P33299|      P40016|      0.69|
|      Q06103|      P40016|      0.81|
|      P38764|      P40016|      0.76|
|      P40016|      P38764|      0.76|
|      P53549|      P40016|      0.55|
|      P40016|      Q08723|       0.7|
|      Q08723|      P40016|       0.7|
|      P40016|      P38886|      0.76|
|      P40016|      P43588|      0.76|
|      P40016|      P38764|      0.76|
|      P40016|      Q06103|      0.81|
|      P53008|      P40016|       0.4|
|      P32565|      P40016|       0.4|
|      P40016|      P32565|       0.4|
|      P38764|      P40016|      0.76|
|      P53549|      P40016|      0.55|
|      P40016|      P38764|      0.76|
|      Q08723|      P40016|       0.7|
+------------+------------+----------+
only showing top 20 rows
```

```
drugBankDF.show()
```

```
+-----------+----------+
|ID_DrugBank|ID_UniProt|
+-----------+----------+
|    DB00001|    P00734|
|    DB00002|    P00533|
|    DB00002|    O75015|
|    DB00002|    P00736|
|    DB00002|    P02745|
|    DB00002|    P02746|
|    DB00002|    P02747|
|    DB00002|    P08637|
|    DB00002|    P09871|
|    DB00002|    P12314|
|    DB00002|    P12318|
|    DB00002|    P31994|
|    DB00002|    P31995|
|    DB00004|    P01589|
|    DB00004|    P14784|
|    DB00004|    P31785|
|    DB00005|    P01375|
|    DB00005|    P20333|
|    DB00005|    P12314|
|    DB00005|    P08637|
+-----------+----------+
only showing top 20 rows
```

Viene, dunque, effettuata l'esportazione del DataFrame ppiDF (utilizzando una apposita funzione della libreria Pandas), utile in seguito per la generazione dei grafi:

```
ppiDF.toPandas().to_csv('ppi.csv', index=False, header=False)
```

Infine, è stato creato un terzo DataFrame, drugTargetsDF, in cui, per ciascuna drug, è presente la lista delle proteine associate. Questo sarà utile per far si che il sistema in seguito possa raccomandare, per ciascuna drug, proteine che non sono presenti in questa lista per quella specifica drug:

```
drugTargetsDF = drugBankDF.groupBy("ID_DrugBank").agg(collect_list("ID_UniProt").alias("Proteins")) \
                          .orderBy('ID_DrugBank')
drugTargetsDF.show()
```

```
+-----------+--------------------+
|ID_DrugBank|            Proteins|
+-----------+--------------------+
|    DB00001|            [P00734]|
|    DB00002|[P00533, O75015, ...|
|    DB00004|[P01589, P14784, ...|
|    DB00005|[P01375, P20333, ...|
|    DB00006|            [P00734]|
|    DB00007|            [P30968]|
|    DB00008|    [P48551, P17181]|
|    DB00009|[P00747, P02671, ...|
|    DB00010|            [Q02643]|
|    DB00011|    [P48551, P17181]|
|    DB00012|            [P19235]|
|    DB00013|[P00747, Q03405, ...|
|    DB00014|    [P22888, P30968]|
|    DB00015|[P00747, P02671, ...|
|    DB00016|            [P19235]|
|    DB00017|            [P30988]|
|    DB00018|    [P48551, P17181]|
|    DB00019|            [Q99062]|
|    DB00020|[P15509, P26951, ...|
|    DB00022|    [P48551, P17181]|
+-----------+--------------------+
only showing top 20 rows
```

2.3 Creazione del DataFrame da usare per il sistema di raccomandazione

In questa sezione viene generato e raffinato il DataFrame che sarà utilizzato dal sistema di raccomandazione. A tale proposito, viene creato un DataFrame, joinedDF, frutto del join tra gli ID_UniProt di drugBankDF e gli Interactor_A di ppiDF, filtrando soltanto quelle proteine di ppiDF che hanno un valore di Confidence >= 0.5:

```
joinedDF = drugBankDF.join(ppiDF, (drugBankDF.ID_UniProt == ppiDF.Interactor_A) & (ppiDF.Confidence >= 0.5))
joinedDF.orderBy('ID_DrugBank').show()
```

```
+-----------+----------+------------+------------+----------+
|ID_DrugBank|ID_UniProt|Interactor_A|Interactor_B|Confidence|
+-----------+----------+------------+------------+----------+
|    DB00001|    P00734|      P00734|  EBI-941456|      0.56|
|    DB00001|    P00734|      P00734|      Q846V4|      0.73|
|    DB00001|    P00734|      P00734|      Q846V4|      0.73|
|    DB00001|    P00734|      P00734|      Q846V4|      0.73|
|    DB00001|    P00734|      P00734|  EBI-941456|      0.56|
|    DB00002|    P00533|      P00533|      P00533|      0.98|
|    DB00002|    P12314|      P12314|    Q9BXN2-6|      0.56|
|    DB00002|    P00533|      P00533|      P62994|      0.56|
|    DB00002|    P31994|      P31994|      P01857|      0.56|
|    DB00002|    P31994|      P31994|      P01857|      0.56|
|    DB00002|    P12314|      P12314|    Q8N6F1-2|      0.56|
|    DB00002|    P31994|      P31994|      P01857|      0.56|
|    DB00002|    P00533|      P00533|      P07948|      0.82|
|    DB00002|    P31994|      P31994|      P01857|      0.56|
|    DB00002|    P00533|      P00533|    P07948-1|      0.54|
|    DB00002|    P12318|      P12318|      Q9UMX0|      0.56|
|    DB00002|    P12314|      P12314|      P07306|      0.56|
|    DB00002|    P00533|      P00533|      P00533|      0.98|
|    DB00002|    P31994|      P31994|      P01857|      0.56|
|    DB00002|    P12314|      P12314|    Q6UX27-3|      0.56|
+-----------+----------+------------+------------+----------+
only showing top 20 rows
```

In questo modo, ciascun record di joinedDF contiene una proteina ID_UniProt = Interactor_A che è associata direttamente con la drug ID_DrugBank, e una proteina Interactor_B che ha una interazione con la proteina ID_UniProt e che potenzialmente potrebbe essere una proteina raccomandata dal sistema per la drug ID_DrugBank, con un valore di Confidence tra ID_UniProt e Interactor_B sufficientemente alto.
Il criterio per il quale raccomandare una proteina Interactor_B per una determinata ID_DrugBank sarà presto spiegato.

A joinedDF viene, dunque, aggiunta la feature della lista delle proteine associate in modo diretto alla drug ID_DrugBank:

```
joinedDF = joinedDF.withColumnRenamed("ID_DrugBank", "ID_Drug")
joinedDF = joinedDF.join(drugTargetsDF, drugTargetsDF.ID_DrugBank == joinedDF.ID_Drug)
joinedDF = joinedDF.select(joinedDF['ID_Drug'], joinedDF['ID_UniProt'], joinedDF['Interactor_A'],
                           joinedDF['Interactor_B'], joinedDF['Proteins'])

joinedDF.orderBy('ID_Drug').show()
```

```
+-------+----------+------------+------------+--------------------+
|ID_Drug|ID_UniProt|Interactor_A|Interactor_B|            Proteins|
+-------+----------+------------+------------+--------------------+
|DB00001|    P00734|      P00734|  EBI-941456|            [P00734]|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|
|DB00001|    P00734|      P00734|  EBI-941456|            [P00734]|
|DB00002|    P00533|      P00533|      P62993|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P06493|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P38646|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P11142|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P22681|[P00533, O75015, ...|
|DB00002|    P09871|      P09871|      P00736|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P31943|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P62993|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      Q06124|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      Q05397|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P29353|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      Q71U36|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P62993|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P10599|[P00533, O75015, ...|
|DB00002|    P00533|      P00533|      P63104|[P00533, O75015, ...|
+-------+----------+------------+------------+--------------------+
only showing top 20 rows
```

La feature Confidence è stata eliminata dalla selezione in quanto non più utile.

In ciascun record, se una proteina Interactor_B dovesse essere presente nella lista Proteins di proteine associate direttamente alla drug ID_Drug, allora questa sarà scartata per il sistema di raccomandazione.

Viene aggiunta a joinedDF una ulteriore feature, Interaction, con tutti i valori settati ad 1, utile in seguito per stabilire il criterio di raccomandazione:

```
joinedDF = joinedDF.withColumn("Interaction", lit(1))
joinedDF.orderBy('ID_Drug').show()
```

```
+-------+----------+------------+------------+--------------------+-----------+
|ID_Drug|ID_UniProt|Interactor_A|Interactor_B|            Proteins|Interaction|
+-------+----------+------------+------------+--------------------+-----------+
|DB00001|    P00734|      P00734|  EBI-941456|            [P00734]|          1|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|          1|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|          1|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|          1|
|DB00001|    P00734|      P00734|  EBI-941456|            [P00734]|          1|
|DB00002|    P00533|      P00533|      P62993|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P06493|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P38646|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P11142|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P22681|[P00533, O75015, ...|          1|
|DB00002|    P09871|      P09871|      P00736|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P31943|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P62993|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      Q06124|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      Q05397|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P29353|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      Q71U36|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P62993|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P10599|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      P63104|[P00533, O75015, ...|          1|
+-------+----------+------------+------------+--------------------+-----------+
only showing top 20 rows
```

Dal momento che joinedDF presenta duplicati, questi vengono eliminati, insieme a quei record che presentano in Interactor_B valori "-":

```
joinedDF = joinedDF.filter(joinedDF.Interactor_B != "-")
joinedDF = joinedDF.dropDuplicates()

joinedDF.orderBy('ID_DrugBank', 'Interactor_B').show()
```

```
+-------+----------+------------+------------+--------------------+-----------+
|ID_Drug|ID_UniProt|Interactor_A|Interactor_B|            Proteins|Interaction|
+-------+----------+------------+------------+--------------------+-----------+
|DB00001|    P00734|      P00734|  EBI-941456|            [P00734]|          1|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|          1|
|DB00002|    P00533|      P00533|      A4FU49|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533| EBI-4399559|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|   NP_059022|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O00170|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O00401|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O00459|[P00533, O75015, ...|          1|
|DB00002|    P12314|      P12314|      O00526|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O00750|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O14543|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O14544|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O14818|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O14944|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O14965|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O15511|[P00533, O75015, ...|          1|
|DB00002|    P12314|      P12314|      O43491|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O43561|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O43639|[P00533, O75015, ...|          1|
|DB00002|    P00533|      P00533|      O43707|[P00533, O75015, ...|          1|
+-------+----------+------------+------------+--------------------+-----------+
only showing top 20 rows
```

Viene, dunque, aggiunta in joinedDF una nuova feature di tipo booleana, Interactor_drug_target, con valore *true* se la proteina Interactor_B è presente nella lista Proteins (e, dunque, ha una associazione diretta con la drug ID_Drug), *false* altrimenti:

```
joinedDF = joinedDF.withColumn("Interactor_drug_target", expr("array_contains(Proteins, Interactor_B)"))
joinedDF.orderBy('ID_DrugBank', 'Interactor_B').show(50)
```

```
+-------+----------+------------+------------+--------------------+-----------+---------------------+
|ID_Drug|ID_UniProt|Interactor_A|Interactor_B|            Proteins|Interaction|Interactor_drug_target|
+-------+----------+------------+------------+--------------------+-----------+---------------------+
|DB00001|    P00734|      P00734|  EBI-941456|            [P00734]|          1|                false|
|DB00001|    P00734|      P00734|      Q846V4|            [P00734]|          1|                false|
|DB00002|    P00533|      P00533|      A4FU49|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533| EBI-4399559|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|   NP_059022|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O00170|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O00401|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O00459|[P00533, O75015, ...|          1|                false|
|DB00002|    P12314|      P12314|      O00526|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O00750|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O14543|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O14544|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O14818|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O14944|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O14965|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O15511|[P00533, O75015, ...|          1|                false|
|DB00002|    P12314|      P12314|      O43491|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O43561|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O43639|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O43707|[P00533, O75015, ...|          1|                false|
|DB00002|    P12314|      P12314|      O43736|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O60603|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O60716|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O60884|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O75095|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O75368|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O75674|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O75791|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O94875|[P00533, O75015, ...|          1|                false|
|DB00002|    P12314|      P12314|      O95393|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O95433|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      O95782|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P00533|[P00533, O75015, ...|          1|                 true|
|DB00002|    P09871|      P09871|      P00736|[P00533, O75015, ...|          1|                 true|
|DB00002|    P00736|      P00736|      P00736|[P00533, O75015, ...|          1|                 true|
|DB00002|    P00533|      P00533|      P01133|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P01135|[P00533, O75015, ...|          1|                false|
|DB00002|    P12314|      P12314|      P01857|[P00533, O75015, ...|          1|                false|
|DB00002|    P31994|      P31994|      P01857|[P00533, O75015, ...|          1|                false|
|DB00002|    P02747|      P02747|      P02745|[P00533, O75015, ...|          1|                 true|
|DB00002|    P02746|      P02746|      P02745|[P00533, O75015, ...|          1|                 true|
|DB00002|    P02745|      P02745|      P02746|[P00533, O75015, ...|          1|                 true|
|DB00002|    P02745|      P02745|      P02747|[P00533, O75015, ...|          1|                 true|
|DB00002|    P00533|      P00533|   P03372-4|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P04083|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P04406|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P04626|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P04792|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P05067|[P00533, O75015, ...|          1|                false|
|DB00002|    P00533|      P00533|      P06493|[P00533, O75015, ...|          1|                false|
+-------+----------+------------+------------+--------------------+-----------+---------------------+
only showing top 50 rows
```

Infine, per stabilire il criterio da adottare per il sistema di raccomandazione, viene effettuato un raggruppamento di tutti quei record aventi medesimo ID_Drug e Interactor_B e Interactor_drug_target settato a *false*, con Interactions contentente il numero dei conteggi di ciascuno di questi raggruppamenti. In questo modo, la feature Interactions verrà usata come rating per il sistema di raccomandazione.
Dunque, per ciascun record, Interactions altro non è che il numero di proteine che sono direttamente associate alla drug ID_Drug che hanno una interazione con la proteina Interactor_B (non associata direttamente alla drug ID_Drug e, dunque, potenziale raccomandata).

```
joinedDF = joinedDF.filter(joinedDF.Interactor_drug_target == 'false').groupBy("ID_Drug","Interactor_B").count()
joinedDF = joinedDF.withColumnRenamed("count", "Interactions")

joinedDF.orderBy('Interactions', ascending=False).show()
```

```
+-------+------------+------------+
|ID_Drug|Interactor_B|Interactions|
+-------+------------+------------+
|DB12010|      P08238|          47|
|DB12010|      Q16543|          19|
|DB12010|      P63104|          18|
|DB12010|      P61981|          15|
|DB12010|      P62993|          14|
|DB09130|      P00533|          13|
|DB12010|      Q12933|          11|
|DB12010|      Q04917|          11|
|DB12010|      P27986|          10|
|DB12010|      P31946|          10|
|DB15035|      P08238|          10|
|DB11638|      P00533|          10|
|DB12010|      P31947|           9|
|DB12010|      P19174|           9|
|DB12010|      P46108|           9|
|DB12010|      P07900|           9|
|DB12010|      P22681|           8|
|DB12010|      P04637|           8|
|DB12010|      Q06124|           8|
|DB12695|      P00533|           8|
+-------+------------+------------+
only showing top 20 rows
```

2.4 Conversione dei valori di tipo String in Indici

Dal momento che i valori da usare per addestrare il modello di raccomandazione possono essere soltanto di tipo *Intero* o *Float* (in questo caso, ID_Drug e Interactor_B sono di tipo *Stringa*), viene utilizzata la funzione *StringIndexer* di PySpark per codificare tutti i valori di tipo stringa. Poiché tale funzione può essere usata una colonna per volta, e poiché gli indici devono poter essere distinti in tutte le colonne, viene utilizzata la funzione *Pipelane* di PySpark per ovviare a questo problema. Viene dunque, creato il DataFrame indexedDF che aggiunge alle feature di joinedDF le feature ID_Drug_Index e Interactor_B_Index utilizzando queste due funzioni:

```
drugIndexer = StringIndexer(inputCol='ID_Drug', outputCol='ID_Drug_index').fit(joinedDF)
proteinIndexer = StringIndexer(inputCol='Interactor_B', outputCol='Interactor_B_index').fit(joinedDF)

pipeline = Pipeline(stages=[drugIndexer, proteinIndexer])

indexedDF = pipeline.fit(joinedDF).transform(joinedDF)

indexedDF.show()
```

```
+-------+------------+------------+-------------+-----------------+
|ID_Drug|Interactor_B|Interactions|ID_Drug_index|Interactor_B_index|
+-------+------------+------------+-------------+-----------------+
|DB11712|      P12956|           1|        329.0|            551.0|
|DB00114|      Q9NVD7|           1|        200.0|           4935.0|
|DB04988|      P04406|           2|         41.0|            256.0|
|DB07529|      O75496|           1|        525.0|            102.0|
|DB00823|      P78362|           1|        150.0|            147.0|
|DB12010|      Q9GZT8|           2|          0.0|           3232.0|
|DB00624|      Q15084|           1|         81.0|            122.0|
|DB00074|      O43491|           1|       1220.0|           1123.0|
|DB14487|    O75182-2|           1|          1.0|           1540.0|
|DB00907|      P30825|           1|        214.0|            736.0|
|DB06870|    Q9BY66-3|           1|        619.0|            317.0|
|DB04573|      Q9BQ39|           1|        102.0|            200.0|
|DB01412|      O15354|           1|       1605.0|            263.0|
|DB07126|      Q00526|           2|        377.0|             74.0|
|DB12267|      P0DQD2|           1|         17.0|           1363.0|
|DB06454|      P47211|           1|       2544.0|            266.0|
|DB01169|      Q92793|           2|         15.0|              7.0|
|DB01108|EBI-11177835|           1|        101.0|            463.0|
|DB06492|      O95967|           1|       2399.0|           1488.0|
|DB00527|      O43303|           1|        864.0|            852.0|
+-------+------------+------------+-------------+-----------------+
only showing top 20 rows
```

2.5 Creazione del modello ALS e fitting dei dati

In questa sezione viene effettuato il training e la valutazione del nostro modello. Vengono provati diversi settaggi per trovare il più piccolo valore di RMSE, usato per testare l'accuracy delle predizioni del nostro modello.

Per l'addestramento, viene dapprima splittato randomicamente indexedDF (80% per il training, 20% per il test), dopodichè è stata usata la funzione *ALS* di PySpark, i cui parametri implementati sono:

- *maxIter*: Numero massimo di iterazioni da eseguire (default: 10);
- *regParam*: Parametro di regolarizzazione. Riduce l'overfitting del modello, il che porta ad una riduzione della varianza nelle stime (default: 0.01);
- *rank*: Dimensioni dei feature vectors da usare. Grandi rank possono portare a modelli migliori, ma sono molto più costosi da calcolare (default: 10);
- *alpha*: Costante usata per calcolare la confidence con dataset impliciti (default: 1.0)
- *coldStartStrategy*: Strategia per gestire nuovi o sconosciuti items/users. Impostandolo su 'drop' si escludono questi item dai risultati (default: nan).

Infine, la funzione *RegressionEvaluator* di PySpark ci permette di valutare il valore di RMSE per i paramentri di regParam, rank e alpha scelti.

```
(training,test) = indexedDF.randomSplit([0.8, 0.2])

regParams = [0.01, 0.1]
ranks = [25]
alphas = [10.0, 20.0, 40.0, 60.0, 80.0]

aus_regParam = 0.0
aus_rank = 0
aus_alpha = 0.0
aus_rmse = 0.0

print('Creating ALS model ...')
for regParam in regParams:
    for rank in ranks:
        for alpha in alphas:
            aus_als = ALS(maxIter=10, regParam=regParam, rank=rank, alpha=alpha, userCol='ID_Drug_index',
                          itemCol="Interactor_B_index", ratingCol="Interactions", coldStartStrategy="drop")
            aus_model = aus_als.fit(training)
            predictions = aus_model.transform(test)
            evaluator = RegressionEvaluator(metricName="rmse", labelCol="Interactions", predictionCol="prediction")
            rmse = evaluator.evaluate(predictions)

            if(aus_rmse == 0.0 or rmse < aus_rmse):
                aus_regParam = regParam
                aus_rank = rank
                aus_alpha = alpha
                aus_rmse = rmse
                model = aus_model

            print("For regParam: {0}, rank: {1}, alpha: {2}, RMSE: {3}".format(regParam, rank, alpha, rmse))

print('Chosen parameters: regParam = {0}, rank = {1}, alpha = {2}'.format(aus_regParam, aus_rank, aus_alpha))
```

```
Creating ALS model ...
For regParam: 0.01, rank: 25, alpha: 10.0, RMSE: 0.2846906755205193
For regParam: 0.01, rank: 25, alpha: 20.0, RMSE: 0.2846906755205193
For regParam: 0.01, rank: 25, alpha: 40.0, RMSE: 0.2846906755205193
For regParam: 0.01, rank: 25, alpha: 60.0, RMSE: 0.2846906755205193
For regParam: 0.01, rank: 25, alpha: 80.0, RMSE: 0.2846906755205193
For regParam: 0.1, rank: 25, alpha: 10.0, RMSE: 0.2689031685224229
For regParam: 0.1, rank: 25, alpha: 20.0, RMSE: 0.2689031685224229
For regParam: 0.1, rank: 25, alpha: 40.0, RMSE: 0.26890316852242296
For regParam: 0.1, rank: 25, alpha: 60.0, RMSE: 0.2689031685224229
For regParam: 0.1, rank: 25, alpha: 80.0, RMSE: 0.26890316852242296
Chosen parameters: regParam = 0.1, rank = 25, alpha = 10.0
```

Il modello creato con una combinazione di regParam, rank e alpha che genera il valore di RMSE più basso sarà quello scelto per il sistema di raccomandazione.

2.6 Esecuzione del Sistema di Raccomandazione

Generato il modello, viene utilizzata la funzione *recommendForAllUsers()* di PySpark che prende a parametro un numero n di proteine che si vuole che il sistema raccomandi per ciascuna drug, e restituisce, per ciascuna ID_Drug_index, n coppie [Interactor_B_index, rating] raccomandate:

```
print("Insert a number of recommendations per drug:")
n = int(input())

protein_recs = model.recommendForAllUsers(n)

protein_recs.show()
```

```
Insert a number of recommendations per drug:
5
+-------------+--------------------+
|ID_Drug_index|     recommendations|
+-------------+--------------------+
|         1580|[[5263, 3.0869117...|
|          471|[[5263, 3.0935702...|
|         1591|[[3543, 3.6073778...|
|         4101|[[5263, 3.1222749...|
|         1342|[[5263, 3.0735283...|
|         2122|[[5263, 3.1680398...|
|         2142|[[5263, 3.194394]...|
|          463|[[5263, 2.9468124...|
|          833|[[5263, 3.0409713...|
|         3794|[[5263, 2.6746826...|
|         1645|[[5263, 3.204765]...|
|         3175|[[5263, 2.935325]...|
|          496|[[3543, 2.74789],...|
|         2366|[[5263, 3.0821824...|
|         2866|[[5263, 3.0592232...|
|         3997|[[3543, 2.924361]...|
|          148|[[5263, 2.983145]...|
|         1088|[[5263, 2.9785137...|
|         1238|[[5263, 3.1817076...|
|         3918|[[5263, 3.1363788...|
+-------------+--------------------+
only showing top 20 rows
```

Infine, viene divisa la colonna recommendations nelle due colonne Interactor_B_index e rating:

```
flatDrugRecs = protein_recs.withColumn('proteinAndRating', explode(protein_recs.recommendations))\
                                        .select('ID_Drug_index', 'proteinAndRating.*')

flatDrugRecs.show()
```

```
+-------------+-----------------+---------+
|ID_Drug_index|Interactor_B_index|   rating|
+-------------+-----------------+---------+
|         1580|             5263|3.0869117|
|         1580|             6119|3.0869117|
|         1580|             3543|2.8562443|
|         1580|             6280|2.3151839|
|         1580|             1137|1.9725999|
|          471|             5263|3.0935702|
|          471|             6119|3.0935702|
|          471|             3543| 2.849777|
|          471|             6280|2.3201783|
|          471|             1137|2.0504289|
|         1591|             3543|3.6073778|
|         1591|             5263|3.2085395|
|         1591|             6119|3.2085395|
|         1591|             6280|2.4064047|
|         1591|             3937|2.0296333|
|         4101|             5263|3.1222749|
|         4101|             6119|3.1222749|
|         4101|             3543|2.8859322|
|         4101|             6280|2.3417063|
|         4101|             1137|1.9761636|
+-------------+-----------------+---------+
only showing top 20 rows
```

2.7 Conversione degli indici nelle stringhe originali

Non essendo intuitivo stabilire a quale ID_Drug si riferisce un valore di ID_Drug_index (rispettivamente a quale Interactor_B si riferisce un valore di Interactor_B_index), vengono utilizzate le funzioni *IndexToString* e *Pipeline* di PySpark per convertire gli indici nelle loro stringe di origine:

```
drugString = IndexToString(inputCol='ID_Drug_index', outputCol='ID_Drug', labels=drugIndexer.labels)
proteinString = IndexToString(inputCol='Interactor_B_index', outputCol='ID_Protein', labels=proteinIndexer.labels)

convertedDrugRecs = Pipeline(stages=[drugString, proteinString]).fit(indexedDF).transform(flatDrugRecs)
convertedDrugRecs = convertedDrugRecs.select(convertedDrugRecs['ID_Drug'], convertedDrugRecs['ID_Protein'],
                                             convertedDrugRecs['rating'])

convertedDrugRecs.select('ID_Drug', 'ID_Protein', 'rating').orderBy('rating', ascending=False).show()
```

```
+-------+----------+---------+
|ID_Drug|ID_Protein|   rating|
+-------+----------+---------+
|DB12010|    P08238|12.920674|
|DB12010|    P63104| 8.531653|
|DB12010|    P61981| 7.272068|
|DB12010|    Q16543|6.8025374|
|DB12010|    Q04917|6.7631745|
|DB08236|    Q9BPX5| 4.915586|
|DB08235|    Q9BPX5|4.8028083|
|DB08515|    Q9Y244|4.7891808|
|DB07728|    Q9Y375| 4.620285|
|DB07728|    Q9P032| 4.620285|
|DB04160|    Q9P032| 4.534795|
|DB04160|    Q9Y375| 4.534795|
|DB08236|    Q9P032| 4.426154|
|DB08236|    Q9Y375| 4.426154|
|DB08358|    Q9P032|4.2863016|
|DB08358|    Q9Y375|4.2863016|
|DB12695|    Q9UQL6|  4.26186|
|DB12695|    P30305|  4.26186|
|DB07080|    Q9Y244|4.1399984|
|DB00162|    Q9P032|4.0918837|
+-------+----------+---------+
only showing top 20 rows
```

2.8 Visualizzazione del Sistema di Raccomandazione

In questa sezione, dato un ID_Drug, vengono visualizzate le n proteine raccomandate per quella specifica ID_Drug e viene creato un file csv 'rec.csv' contenenti questi record, utile in seguito per la generazione dei grafi. Inoltre, viene affiancata anche la visualizzazione di joinedDF filtrata con ID_Drug uguale alla ID_Drug inserita in ordine discendente di Interactions. In questo modo è possibile stabilire se le proteine predette sono effettivamente quelle con i valori di Interactions più alti.

```
print("Insert an id_drug: ")
id_drug = input()

print('Recommended Proteins for {0}'.format(id_drug))
convertedDrugRecs.filter(convertedDrugRecs.ID_Drug.isin(id_drug)).select(convertedDrugRecs['ID_Protein'],
                                                                         convertedDrugRecs['rating']).show(n)

csvDF = convertedDrugRecs.filter(convertedDrugRecs.ID_Drug.isin(id_drug)).select(convertedDrugRecs['ID_Protein'],
                                                                                 convertedDrugRecs['rating'])
csvDF.toPandas().to_csv('rec.csv', index=False)

print('joinedDF')
joinedDF.filter(joinedDF.ID_Drug.isin(id_drug)).orderBy('Interactions', 'Interactor_B', ascending=False).show()
```

```
Insert an id_drug:
DB12010
Recommended Proteins for DB12010
+----------+---------+
|ID_Protein|   rating|
+----------+---------+
|    P08238|12.920674|
|    P63104| 8.531653|
|    P61981| 7.272068|
|    Q16543|6.8025374|
|    Q04917|6.7631745|
+----------+---------+

joinedDF
+-------+------------+------------+
|ID_Drug|Interactor_B|Interactions|
+-------+------------+------------+
|DB12010|      P08238|          47|
|DB12010|      Q16543|          19|
|DB12010|      P63104|          18|
|DB12010|      P61981|          15|
|DB12010|      P62993|          14|
|DB12010|      Q12933|          11|
|DB12010|      Q04917|          11|
|DB12010|      P31946|          10|
|DB12010|      P27986|          10|
|DB12010|      P46108|           9|
|DB12010|      P31947|           9|
|DB12010|      P19174|           9|
|DB12010|      P07900|           9|
|DB12010|      Q06124|           8|
|DB12010|      P22681|           8|
|DB12010|      P04637|           8|
|DB12010|      P61962|           7|
|DB12010|      P60953|           7|
|DB12010|      P40763|           7|
|DB12010|      Q7Z3S9|           6|
+-------+------------+------------+
only showing top 20 rows
```

2.9 Generazione e visualizzazione dei Grafi

In questa sezione, vengono generati i grafi per un'analisi più accurata e visualmente più intuitiva dei risultati.

Prima di tutto, dal file "rec.csv" creato in precedenza contenente le proteine raccomandate dal sistema per una specifica drug, viene generata una lista *rec_list* contenente queste proteine:

```
rec_list = []

rec_prot_DF = spark.read.csv("rec.csv", header=True, inferSchema=True)
rec_prot_array = np.array(rec_prot_DF.select("ID_Protein").collect())

for rec in rec_prot_array:
    rec_list.append(rec[0])
```

Allo stesso modo, viene creata una lista *dir_list* contenente le proteine che interagiscono in modo diretto con la drug inserita per il sistema di raccomandazione:

```
dir_list = []

dir_prot_DF = drugBankDF.filter(drugBankDF.ID_DrugBank.isin(id_drug))
dir_prot_array = np.array(dir_prot_DF.select("ID_UniProt").collect())

for dir in dir_prot_array:
    dir_list.append(dir[0])
```

Sfruttando, dunque, la libreria NetworkX, mediante le funzioni *DiGraph()* e *add_edge()* viene generato il grafo G contente tutte le PPI prese dal file precedentemente creato "ppi.csv":

```
G = nx.DiGraph()

f = open("ppi.csv", "r")

for line in f:
    node1, node2, weight = line.split(",")
    G.add_edge(node1, node2, weight=float(weight))
```

Il grafo viene dunque filtrato in modo tale da contenere soltanto i nodi collegati con una specifica proteina (raccomandata dal sistema) ad una certa distanza. Per fare questo, è stata usata la funzione *ego_graph* della libreria NetworkX, che prende come parametri il grafo di riferimento, il nodo centrale ed il raggio. Quindi, viene creata una nuova lista, *nodes_list*, contenente questi nodi con il loro relativo peso.

```
print("Insert the center node ")
node = input()
print("Insert the radius ")
radius = input()
G = nx.generators.ego_graph(G, node, radius=int(radius))

nodes_list = []

for line in G.edges():
    nodes_list.append([line[0], line[1], G.edges[line[0], line[1]]["weight"]])
```

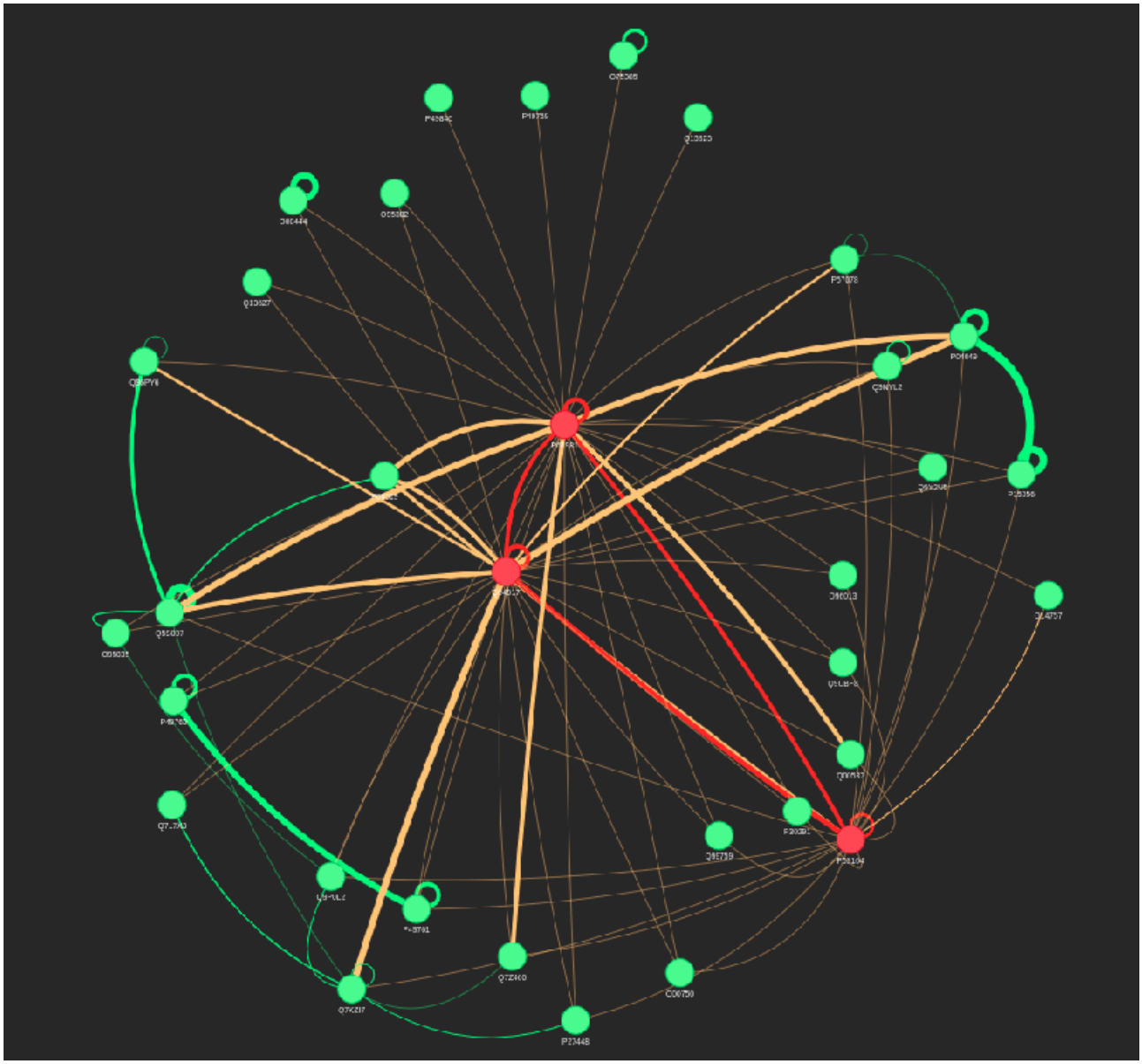
```
Insert the center node
P61981
Insert the radius
1
```

Per la rappresentazione dei grafi interattivi, sono state utilizzate opportune funzioni della libreria PyVis. In particolare, per una data proteina raccomandata, sono stati generati due tipi di grafi: uno completo, contenente tutte le interazioni di questa proteina, ed uno parziale, contenente soltanto le interazioni di questa proteina con le proteine che sono associate direttamente alla drug di riferimento ed eventuali interazioni con altre proteine raccomandate. Questo perchè nella rete PPI ciascuna proteina può avere tantissime interazioni, e spesso una rappresentazione completa del grafo, seppur interattiva, non è molto intuitiva.

```
def partial_graph(nodes_list):
    net = Network(height="100%", width="100%", bgcolor="#222222", font_color="white")

    for i in range(len(nodes_list)):
        node1 = nodes_list[i][0]
        node2 = nodes_list[i][1]
        w = float(nodes_list[i][2])

        if node1 in rec_list:
            if node2 in rec_list:
                net.add_node(node1, color="#ff4d4d")
                net.add_node(node2, color="#ff4d4d")
                net.add_edge(node1, node2, value=w, title=w, color="#ff3300", width=float(w))
            elif node2 in dir_list:
                net.add_node(node1, color="#ff4d4d")
                net.add_node(node2, color="#80ff80")
                net.add_edge(node1, node2, title=w, color="#ffcc66", width=float(w))
            else:
                continue
        elif node1 in dir_list:
            if node2 in rec_list:
                net.add_node(node1, color="#80ff80")
                net.add_node(node2, color="#ff4d4d")
                net.add_edge(node1, node2, value=w, title=w, color="#ffcc66", width=float(w))
            elif node2 in dir_list:
                net.add_node(node1, color="#80ff80")
                net.add_node(node2, color="#80ff80")
                net.add_edge(node1, node2, value=w, title=w, color="#66ff66", width=float(w))
            else:
                continue
        else:
            continue

    #net.show_buttons(filter_=['physics'])
    #net.show_buttons(filter_=['nodes'])

    net.set_options(
    """
    var options = {
      "physics": {
        "forceAtlas2Based": {
          "gravitationalConstant": -268,
          "centralGravity": 0.025,
          "springLength": 265,
          "springConstant": 0.14,
          "damping": 0.17
        },
        "maxVelocity": 0,
        "minVelocity": 0.01,
        "solver": "forceAtlas2Based",
        "timestep": 0.01
      },
      "nodes": {
        "borderWidthSelected": 4
      }
    }
    """
    )

    net.show("{0}_partial_network_for_{1}_IDdrug.html".format(node ,id_drug))
```

```
def complete_graph(nodes_list):
    net = Network(height="100%", width="100%", bgcolor="#222222", font_color="white")

    for i in range(len(nodes_list)):
        node1 = nodes_list[i][0]
        node2 = nodes_list[i][1]
        w = float(nodes_list[i][2])

        if node1 in rec_list:
            if node2 in rec_list:
                net.add_node(node1, color="#ff4d4d")
                net.add_node(node2, color="#ff4d4d")
                net.add_edge(node1, node2, value=w, title=w, color="#ff3300", width=float(w))
            elif node2 in dir_list:
                net.add_node(node1, color="#ff4d4d")
                net.add_node(node2, color="#80ff80")
                net.add_edge(node1, node2, title=w, color="#ffcc66", width=float(w))
            else:
                net.add_node(node1, color="#ff4d4d")
                net.add_node(node2)
                net.add_edge(node1, node2, value=w, title=w, color='black', width=float(w))
        elif node1 in dir_list:
            if node2 in rec_list:
                net.add_node(node1, color="#80ff80")
                net.add_node(node2, color="#ff4d4d")
                net.add_edge(node1, node2, value=w, title=w, color="#ffcc66", width=float(w))
            elif node2 in dir_list:
                net.add_node(node1, color="#80ff80")
                net.add_node(node2, color="#80ff80")
                net.add_edge(node1, node2, value=w, title=w, color="#66ff66", width=float(w))
            else:
                net.add_node(node1, color="#80ff80")
                net.add_node(node2)
                net.add_edge(node1, node2, value=w, title=w, color='black', width=float(w))
        else:
            if node2 in rec_list:
                net.add_node(node1)
                net.add_node(node2, color="#ff4d4d")
                net.add_edge(node1, node2, value=w, title=w, color='black', width=float(w))
            elif node2 in dir_list:
                net.add_node(node1)
                net.add_node(node2, color="#80ff80")
                net.add_edge(node1, node2, value=w, title=w, color='black', width=float(w))
            else:
                net.add_node(node1)
                net.add_node(node2)
                net.add_edge(node1, node2, value=w, title=w, color='black', width=float(w))

    #net.show_buttons(filter_=['physics'])
    #net.show_buttons(filter_=['nodes'])

    net.set_options(
    """
    var options = {
      "physics": {
        "forceAtlas2Based": {
          "gravitationalConstant": -268,
          "centralGravity": 0.025,
          "springLength": 265,
          "springConstant": 0.14,
          "damping": 0.17
        },
        "maxVelocity": 0,
        "minVelocity": 0.01,
        "solver": "forceAtlas2Based",
        "timestep": 0.01
      },
      "nodes": {
        "borderWidthSelected": 4
      }
    }
    """
    )

    net.show("{0}_complete_network_for_{1}_IDdrug.html".format(node ,id_drug))
```

A seguire, un esempio di grafo avente come nodo centrale la proteina P61981 raccomandata dal sistema per la drug DB12010, con raggio impostato ad 1 (valore consigliato):

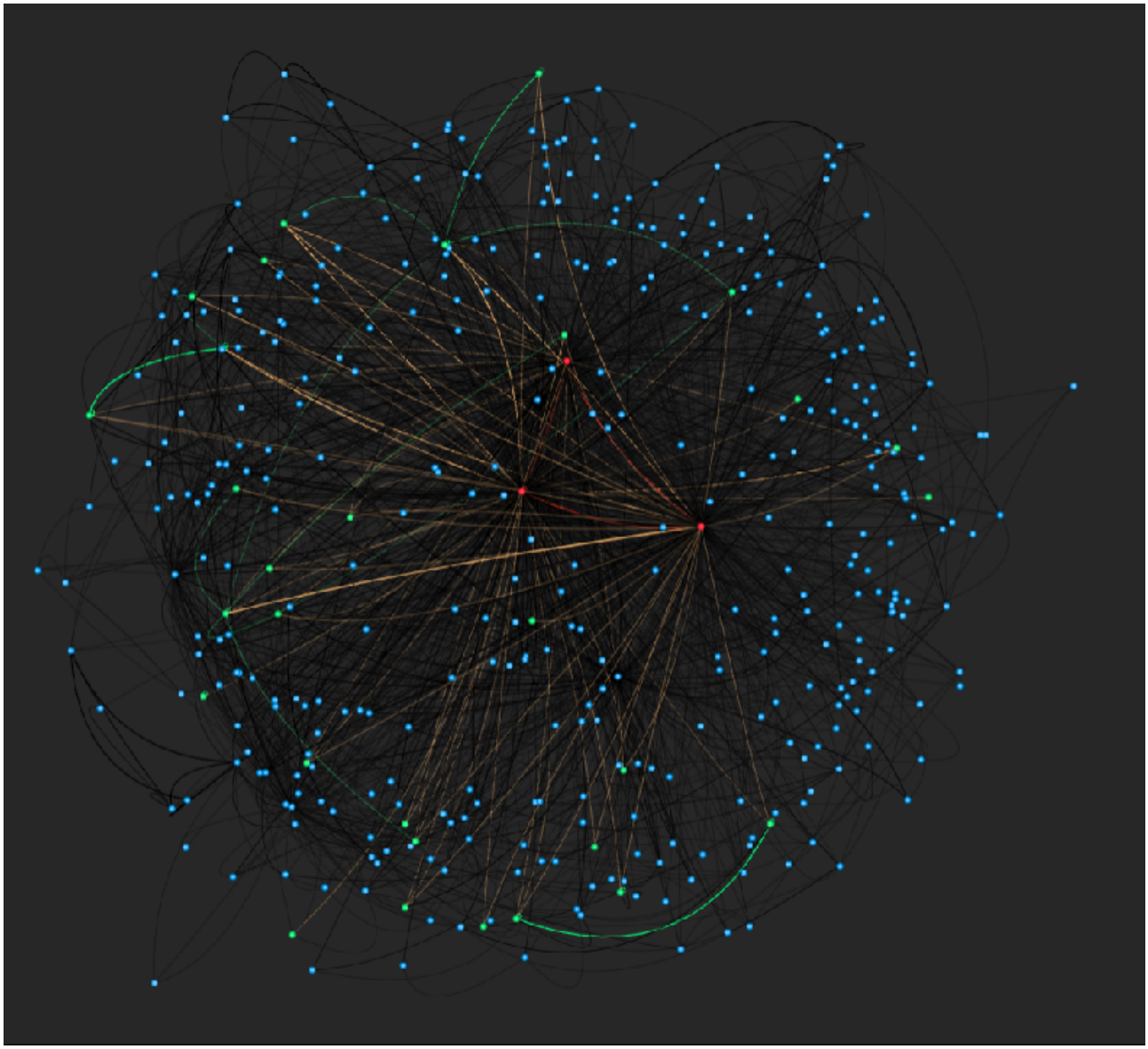


In rosso sono riportate le proteine consigliate dal sistema per la drug DB12010 (da questo esempio, infatti, si vede come la proteina P61981 ha interazione con altre due proteine raccomandate dal sistema), mentre in verde sono rappresentate le proteine che hanno una associazione diretta con la drug DB12010 e che sono collegate allo stesso tempo con la proteina P61981. La dimensione degli archi si riferisce al valore di confidenza tra le due proteine: più è grande questo valore, maggiore sarà la dimensione dell'arco (portando, inoltre, il puntatore del mouse nell'arco desiderato, è possibile visualizzare il valore effettivo di confidenza per quell'arco); gli archi colorati di rosso sono gli archi che mettono in relazione due proteine raccomandate, gli archi colorati in arancione mettono in relazione le proteine raccomandate con le proteine che hanno una associazione diretta con la drug, infine, gli archi colorati in verde mettono in relazione due proteine associate direttamente con la drug.

In questo specifico esempio, le tre proteine raccomandate sono collegate tra loro. In particolare P61981 è collegata con Q04917 (confidence = 0.64) e con P63104 (confidence =

0.64), infine Q04917 è collegata con P63104 (confidence = 0.73). Dal momento che tre proteine raccomandate dal sistema interagiscono tra loro nella rete PPI con valori di confidence piuttosto elevati e dal momento che interagiscono in comune (con anche valori di confidence elevati) con numerose proteine associate direttamente alla drug, è possibile concludere che queste proteine raccomandate sono effettivamente delle ottime candidate per la drug DB12010.

Di seguito, viene mostrato il grafo completo avente sempre come nodo centrale la proteina P61981 raccomandata dal sistema per la drug DB12010, con raggio settato ad 1:



In questo grafo, in blu sono rappresentate tutte le proteine della rete PPI che hanno una relazione con la proteina centrale inserita, ed eventuali altre relazioni. Tali interazioni sono state rappresentate con un arco di colore nero.

3 Conclusioni

Nel realizzare questo progetto è stata analizzata la rete PPI dell'uomo e sono state considerate le associazioni tra drugs e targets. È stato, dunque, costruito un sistema di raccomandazione tale che, in funzione a come le proteine sono collegate sulla rete PPI, data una drug suggesce nuovi target.
Dai risultati ottenuti, mostrati nei grafi interattivi implementati per questo lavoro, è possibile stabilire come le proteine suggerite dal sistema possono essere delle ottime candidate per una data drug.