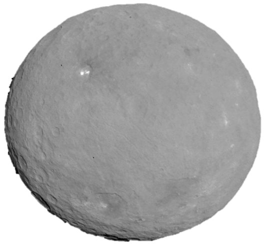
Progetto Data Mining

Analisi “Small-body” database: Asteroidi e Comete

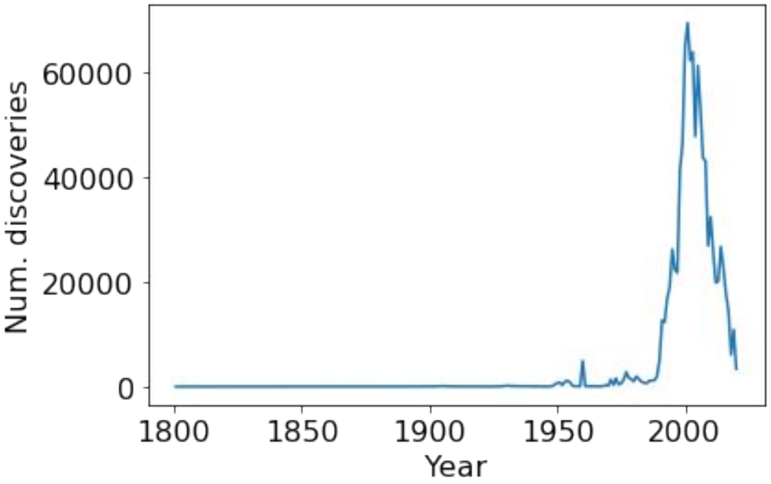
Francesco Pasceri 204963

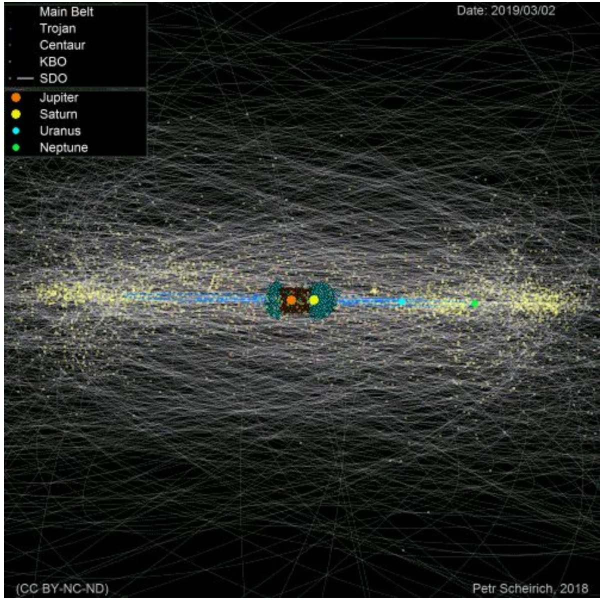
Contesto

Nel 1801, a Palermo, fu scoperto il primo asteroide "*Cerere*"...



Ed oggi...





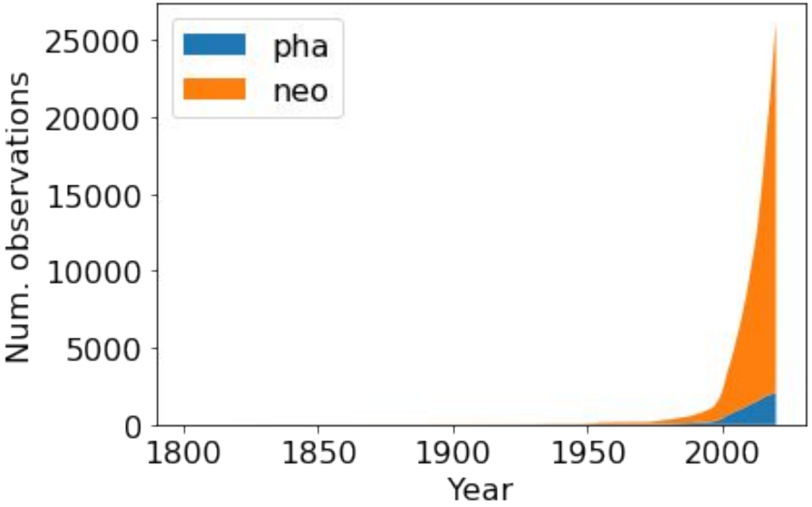
Contesto

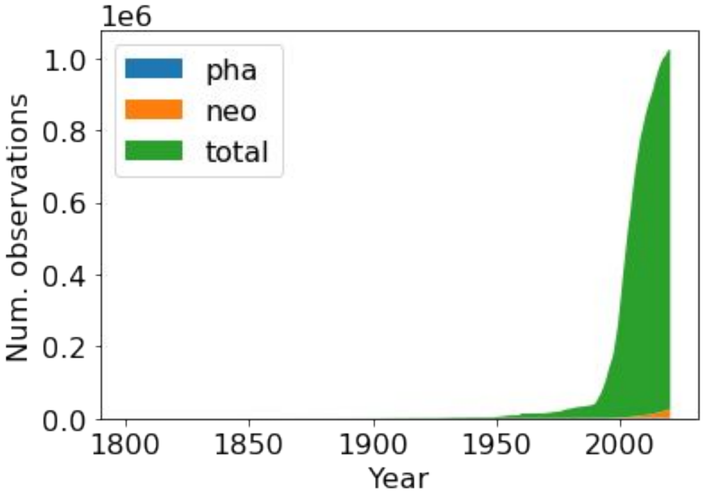
Corpi più presenti nel sistema solare

- Meccanica e dinamica celeste
- Buon campione statistico
- Contengono informazioni storiche
- Possibili pianeti nani
- Potenzialmente disastrosi

Asteroidi → meteore e meteoriti...

Comete





Database e obiettivi

JPL Small-body Database (https://ssd.jpl.nasa.gov/sbdb_query.cgi#x)

	full_name	neo	pha	H	G	M1	diameter	albedo	rot_per	GM	e	a	q	i	om	w	ma	ad	per_y	moid	moid_jup	t_jup	class	first_obs	n_obs_used	condition_code
0	1 Ceres (A801 AA)	N	N	3.40	0.12	NaN	939.400	0.0900	9.074170	62.6284	0.076009	2.769165	2.558684	10.594067	80.305531	73.597695	77.372098	2.979647	4.608202	1.594780	2.09753	3.309	MBA	1995-01-05	1030.0	0
1	2 Pallas (A802 FA)	N	N	4.20	0.11	NaN	545.000	0.1010	7.813200	14.3000	0.229972	2.773841	2.135935	34.832932	173.024741	310.202392	144.975675	3.411748	4.619880	1.234290	1.85093	3.042	MBA	1804-08-27	8477.0	0
2	3 Juno (A804 RA)	N	N	5.33	0.32	NaN	246.596	0.2140	7.210000	NaN	0.256936	2.668285	1.982706	12.991043	169.851482	248.066193	125.435355	3.353865	4.358696	1.034290	2.18899	3.299	MBA	1804-10-17	7188.0	0
3	4 Vesta (A807 FA)	N	N	3.00	0.32	NaN	525.400	0.4228	5.342128	17.8000	0.088721	2.361418	2.151909	7.141771	103.810804	150.728541	95.861938	2.570926	3.628837	1.139480	2.46988	3.535	MBA	1950-09-23	9397.0	0
4	5 Astraea (A845 XA)	N	N	6.90	NaN	NaN	106.699	0.2740	16.806000	NaN	0.190913	2.574037	2.082619	5.367427	141.571026	358.648418	17.846343	3.065455	4.129814	1.095750	1.95968	3.396	MBA	1845-12-15	3034.0	0
5	6 Hebe (A847 NA)	N	N	5.80	0.24	NaN	185.180	0.2679	7.274500	NaN	0.203219	2.424533	1.931822	14.739653	138.643432	239.736273	190.686496	2.917243	3.775290	0.973673	2.63933	3.439	MBA	1848-09-05	6134.0	0
6	7 Iris (A847 PA)	N	N	5.60	NaN	NaN	199.830	0.2766	7.139000	NaN	0.230145	2.387375	1.837933	5.521598	259.563942	145.201546	247.425812	2.936818	3.688835	0.850693	2.47160	3.492	MBA	1848-08-23	5218.0	0
7	8 Flora (A847 UA)	N	N	6.50	0.28	NaN	147.491	0.2260	12.865000	NaN	0.155833	2.201415	1.858362	5.889081	110.876524	285.458915	315.318013	2.544467	3.266337	0.875980	2.87060	3.642	MBA	1847-10-27	2775.0	0
8	9 Metis (A848 HA)	N	N	6.30	0.17	NaN	190.000	0.1180	5.079000	NaN	0.123300	2.386189	2.091972	5.576494	68.909459	6.337325	23.912205	2.680407	3.686087	1.107110	2.55296	3.518	MBA	1849-07-31	2746.0	0
9	10 Hygiea (A849 GA)	N	N	5.50	NaN	NaN	407.120	0.0717	27.630000	7.0000	0.112117	3.142435	2.790114	3.831786	283.198444	312.412932	222.850543	3.494757	5.570674	1.780300	1.53545	3.197	MBA	1849-05-26	3547.0	0

- Calcolo del diametro con modelli di regressione
- Classificazione fascia orbitale
- Anomaly detection *“PHA”*

Analisi database

- Analisi degli attributi

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999016 entries, 0 to 999015
Data columns (total 26 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   full_name       999016 non-null  object
 1   neo             995518 non-null  object
 2   pha             955081 non-null  object
 3   H               989524 non-null  float64
 4   G               119 non-null     float64
 5   M1              1449 non-null    float64
 6   diameter        140299 non-null  float64
 7   albedo          139035 non-null  float64
 8   rot_per         23110 non-null   float64
 9   GM              14 non-null      float64
 10  e               999016 non-null  float64
 11  a               997187 non-null  float64
 12  q               999016 non-null  float64
 13  i               999016 non-null  float64
 14  om              999016 non-null  float64
 15  w               999016 non-null  float64
 16  ma              997186 non-null  float64
 17  ad              996797 non-null  float64
 18  per_y           996654 non-null  float64
 19  moid            956653 non-null  float64
 20  moid_jup        955923 non-null  float64
 21  t_jup           996876 non-null  float64
 22  class           999016 non-null  object
 23  first_obs       998991 non-null  object
 24  n_obs_used      998945 non-null  float64
 25  condition_code  995513 non-null  object
dtypes: float64(20), object(6)
memory usage: 198.2+ MB
```

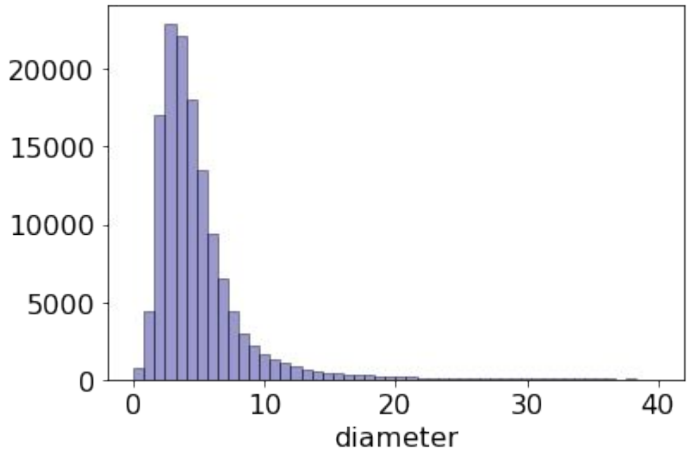
	neo	pha	H	diameter	albedo	e	a	q	i	om	w	ma	ad	n	per_y	moid	moid_jup	t_jup	class	n_obs_used	condition_code
neo	1	0.52	0.27	-0.05	0.03	0.41	-0.06	-0.28	0.12	0	0	-0.01	-0.01	0.47	-0.01	-0.24	0.1	0.35	0.61	-0.04	0.24
pha	0.52	1	0.14	-0.02	0.02	0.21	-0.04	-0.15	0.03	-0	-0	-0	-0.01	0.31	-0.01	-0.13	0.07	0.25	0.37	-0.02	0.06
H	0.27	0.14	1	-0.57	-0.22	0.25	-0.13	-0.39	-0.05	0	-0.01	0.01	-0.07	0.32	-0.04	-0.38	0.21	0.27	-0.01	-0.77	0.22
diameter	-0.05	-0.02	-0.57	1	-0.11	-0.06	0.15	0.33	0.06	0	0	-0	0.09	-0.19	0.05	0.33	-0.11	-0.16	0.14	0.38	-0.03
albedo	0.03	0.02	-0.22	-0.11	1	-0.02	-0.12	-0.27	-0.07	0	-0	-0	-0.08	0.35	-0.02	-0.28	0.38	0.36	0	0.44	-0.05
e	0.41	0.21	0.25	-0.06	-0.02	1	0.01	-0.54	0.17	0	0.01	-0.01	0.11	0.23	0.05	-0.51	-0.03	0.06	0.21	-0.1	0.18
a	-0.06	-0.04	-0.13	0.15	-0.12	0.01	1	0.37	0.16	-0	-0	0.01	0.99	-0.28	0.94	0.38	-0.19	-0.35	0.18	-0.06	0.03
q	-0.28	-0.15	-0.39	0.33	-0.27	-0.54	0.37	1	0.09	-0	-0.01	0.05	0.21	-0.69	0.11	1	-0.51	-0.57	0.24	-0.1	-0.06
i	0.12	0.03	-0.05	0.06	-0.07	0.17	0.16	0.09	1	-0.01	-0.01	0.01	0.15	-0.12	0.1	0.13	-0.12	-0.34	0.19	-0.18	0.09
om	0	-0	0	0	0	0	-0	-0	-0.01	1	-0.11	0	0	0.01	0	-0	0.02	0.01	0.01	-0.02	0.01
w	0	-0	-0.01	0	-0	0.01	-0	-0.01	-0.01	-0.11	1	-0	-0	0	-0	-0.01	0.02	0	0	0.01	-0
ma	-0.01	-0	0.01	-0	-0	-0.01	0.01	0.05	0.01	0	-0	1	-0	-0.03	-0.01	0.05	-0.07	-0.03	0.04	-0.05	0.02
ad	-0.01	-0.01	-0.07	0.09	-0.08	0.11	0.99	0.21	0.15	0	-0	-0	1	-0.16	0.97	0.21	-0.11	-0.27	0.15	-0.05	0.04
n	0.47	0.31	0.32	-0.19	0.35	0.23	-0.28	-0.69	-0.12	0.01	0	-0.03	-0.16	1	-0.06	-0.67	0.71	0.95	0.13	0.15	0.06
per_y	-0.01	-0.01	-0.04	0.05	-0.02	0.05	0.94	0.11	0.1	0	-0	-0.01	0.97	-0.06	1	0.11	-0.01	-0.15	0.08	-0.01	0.02
moid	-0.24	-0.13	-0.38	0.33	-0.28	-0.51	0.38	1	0.13	-0	-0.01	0.05	0.21	-0.67	0.11	1	-0.51	-0.57	0.28	-0.11	-0.05
moid_jup	0.1	0.07	0.21	-0.11	0.38	-0.03	-0.19	-0.51	-0.12	0.02	0.02	-0.07	-0.11	0.71	-0.01	-0.51	1	0.71	-0.24	0.2	-0.03
t_jup	0.35	0.25	0.27	-0.16	0.36	0.06	-0.35	-0.57	-0.34	0.01	0	-0.03	-0.27	0.95	-0.15	-0.57	0.71	1	0.08	0.19	0
class	0.61	0.37	-0.01	0.14	0	0.21	0.18	0.24	0.19	0.01	0	0.04	0.15	0.13	0.08	0.28	-0.24	0.08	1	-0.05	0.18
n_obs_used	-0.04	-0.02	-0.77	0.38	0.44	-0.1	-0.06	-0.1	-0.18	-0.02	0.01	-0.05	-0.05	0.15	-0.01	-0.11	0.2	0.19	-0.05	1	-0.12
condition_code	0.24	0.06	0.22	-0.03	-0.05	0.18	0.03	-0.06	0.09	0.01	-0	0.02	0.04	0.06	0.02	-0.05	-0.03	0	0.18	-0.12	1

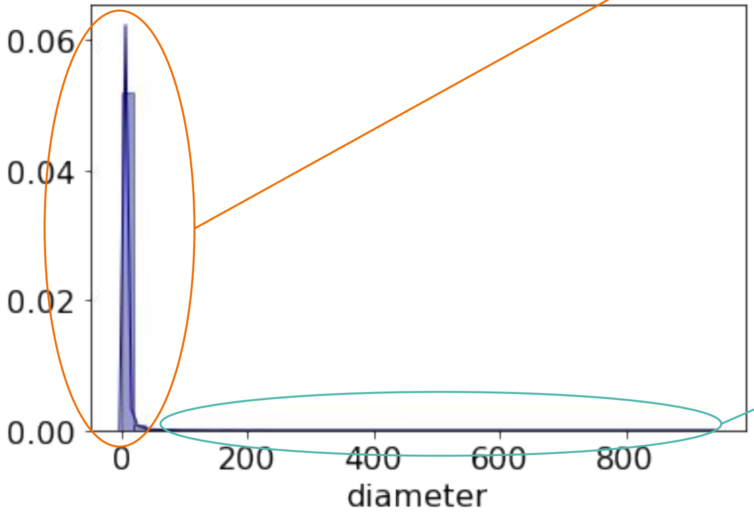
Analisi database

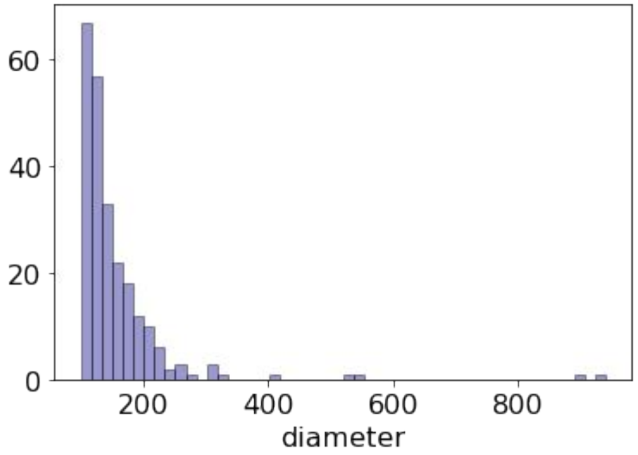
Attributo di interesse:

1. Diametro [km]

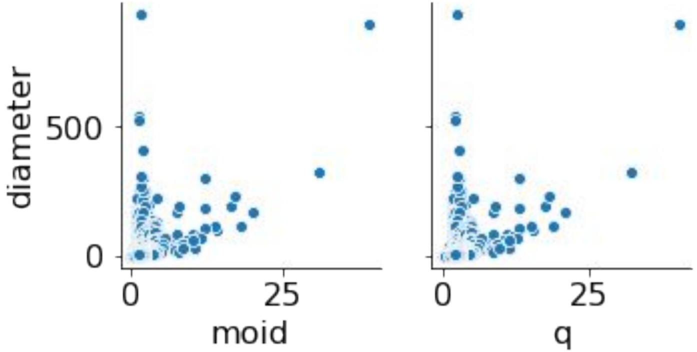
Pochi molto **grandi**.
Molti piccoli.





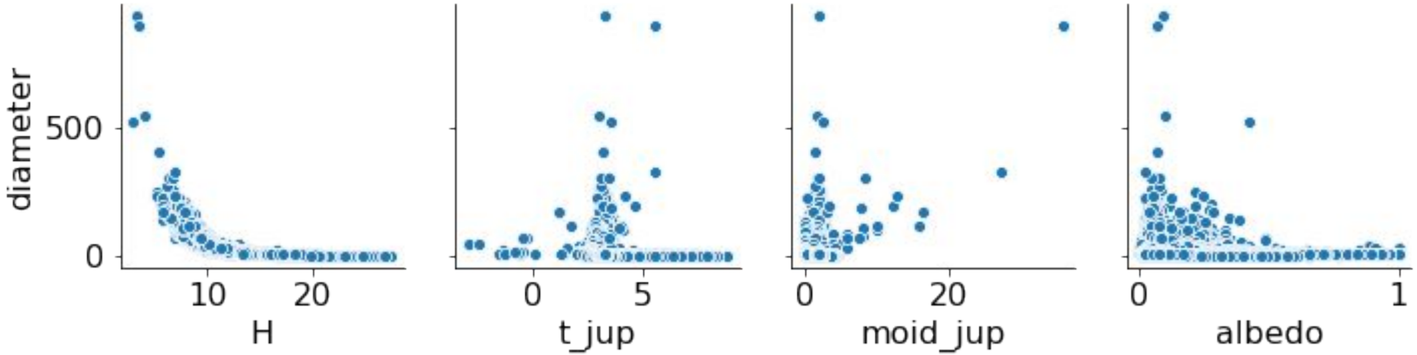


Analisi database



Correlazione positiva?

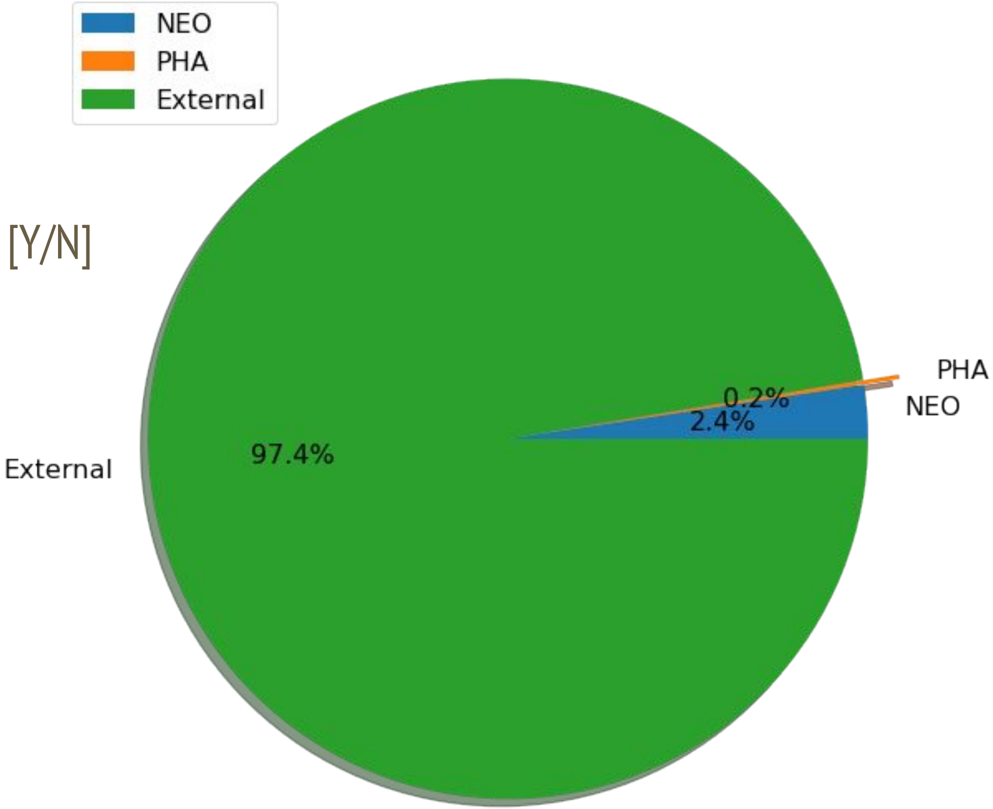
Correlazione negativa?



Analisi database

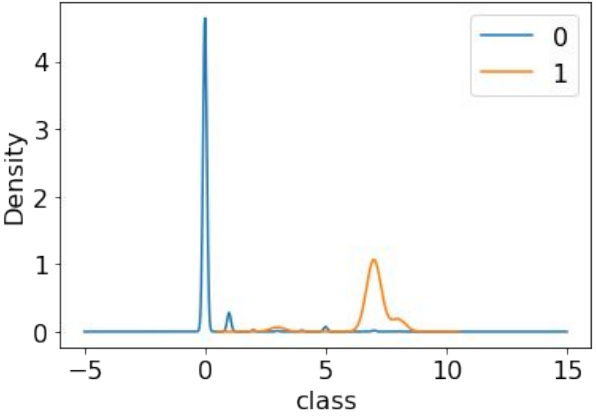
Attributo di interesse:

2. Potentially Hazardous Asteroids (PHAs) [Y/N]



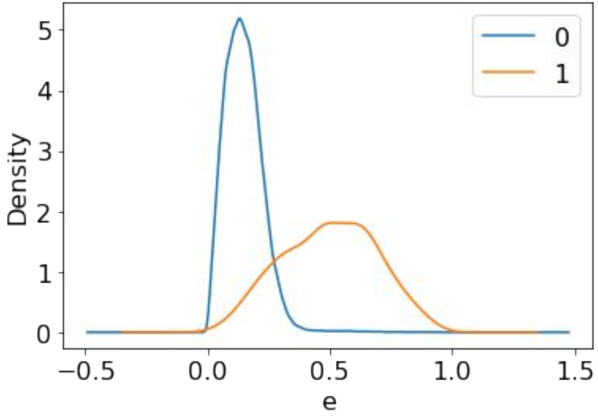
Future missioni per deviare i *PHA*

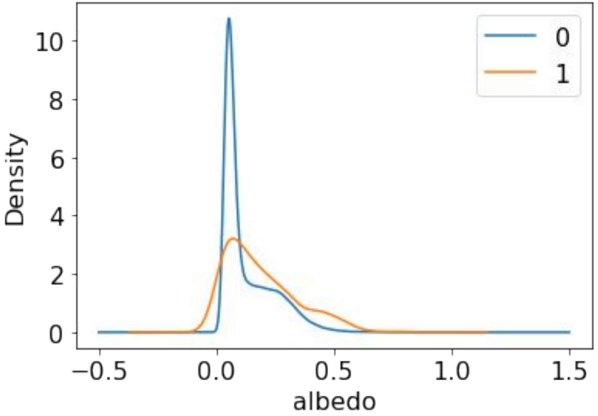
Analisi database



Orbite vicine e circolari

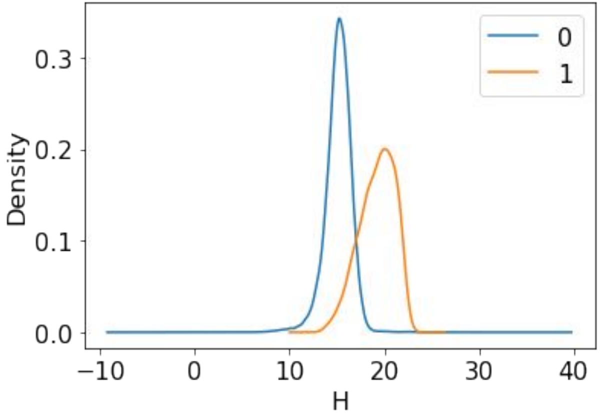
Orbite più lontane ed ellittiche





La riflessione non aiuta la distinzione

Più luminose potrebbero preoccupare?

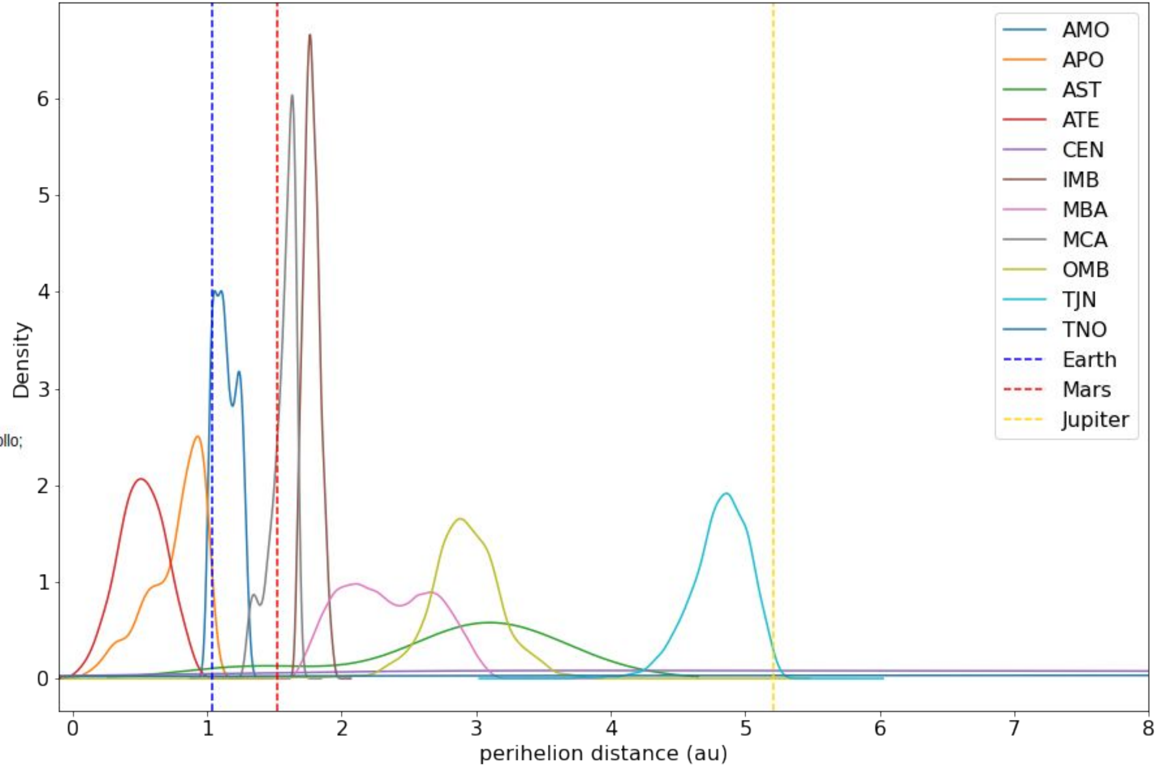


Analisi database

Attributo di interesse:

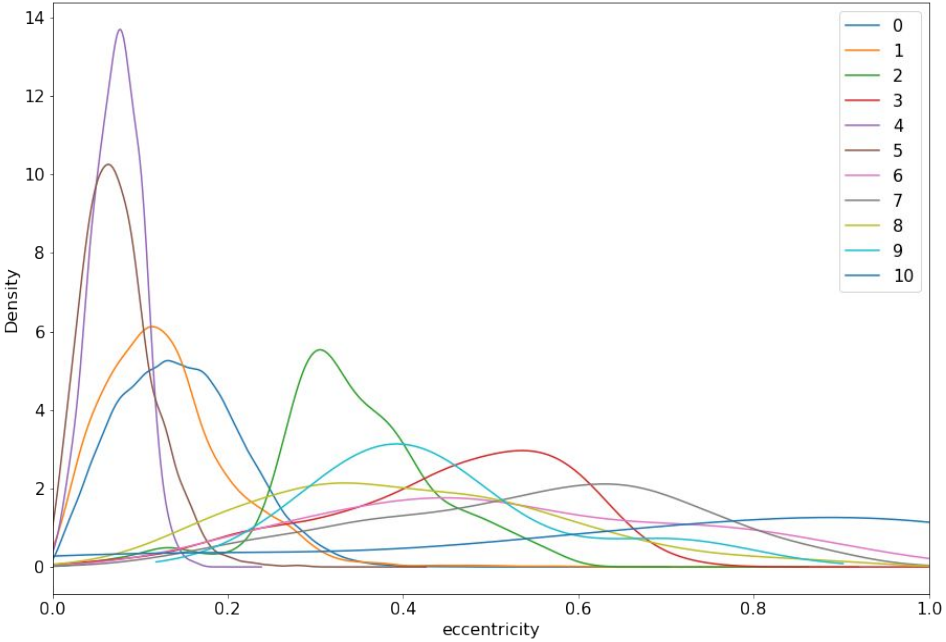
3. Classe orbitale

- 0 MBA -> Main Belt Asteroid : Asteroids with orbital elements;
- 1 OMB -> Outer Main Belt : Asteroids with orbital elements;
- 2 MCA -> Mars Crossing Asteroid : Asteroids that cross the orbit of Mars;
- 3 AMO -> Amor : Near-Earth asteroid orbits similar to that of 1221 Amor;
- 4 IMB -> Inner Main Belt : Asteroids with orbital elements;
- 5 TJN -> Jupiter Trojan : Asteroids trapped in Jupiter's L4/L5 Lagrange points;
- 6 CEN -> Centaur : Objects with orbits between Jupiter and Neptune;
- 7 APO -> Apollo : Near-Earth asteroid orbits which cross the Earth's orbit similar to that of 1862 Apollo;
- 8 ATE -> Atene : Near-Earth asteroid orbits similar to that of 2062 Aten;
- 9 AST -> Asteroid : Asteroid orbit not matching any defined orbit class;
- 10 TNO -> TransNeptunian Object : Objects with orbits outside Neptune;
- 11 ETc -> Encke-type Comet : Encke-type comet;
- 12 COM -> Comete : Comet orbit not matching any defined orbit class;



La distanza del perielio è un attributo con una elevata affinità con la classe orbitale

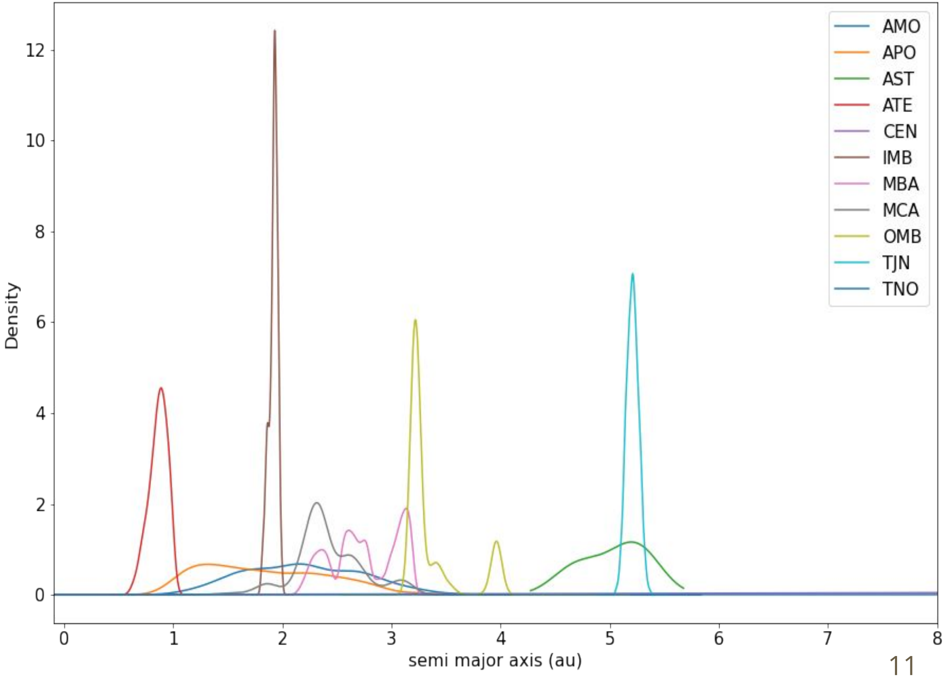
Analisi database



Corpi tra il Sole e Giove hanno orbite circolari.

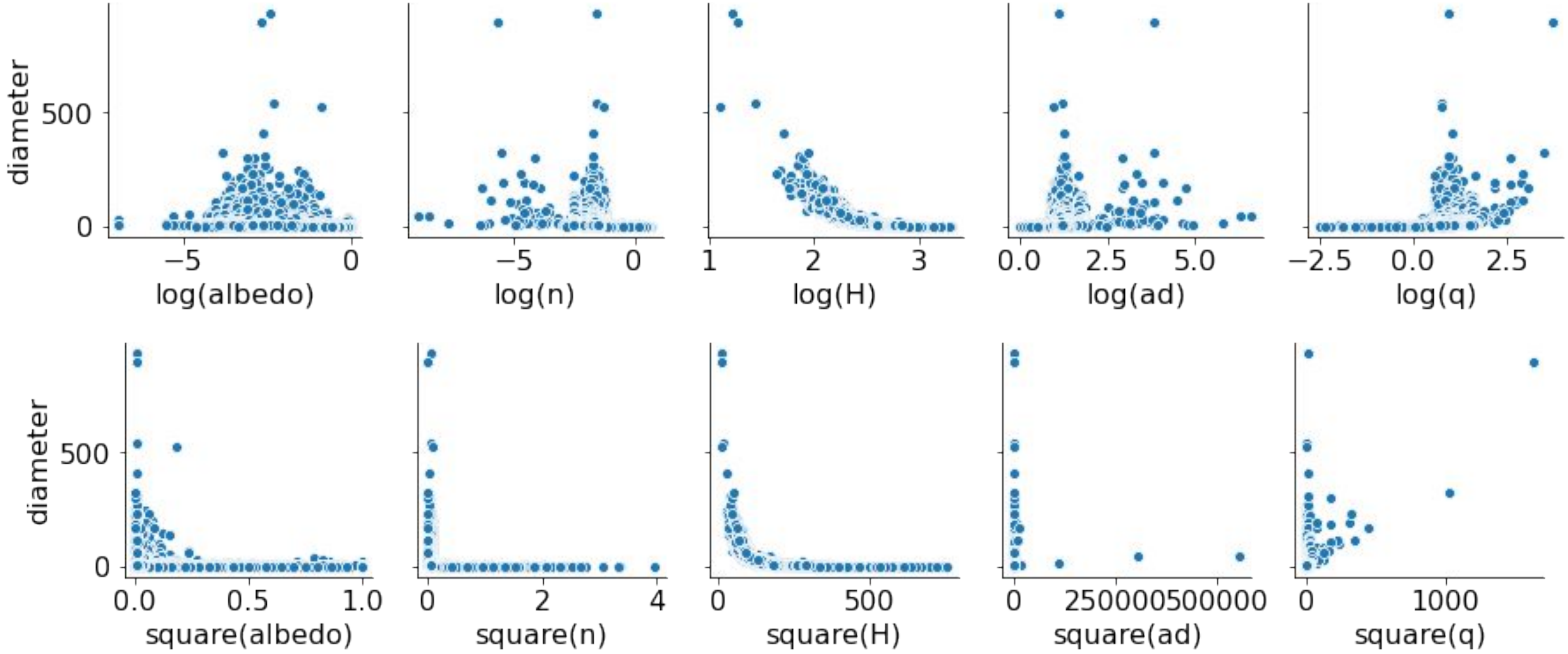
Corpi esterni hanno orbite ellittiche.

Il semiasse maggiore caratterizza lievemente le orbite.

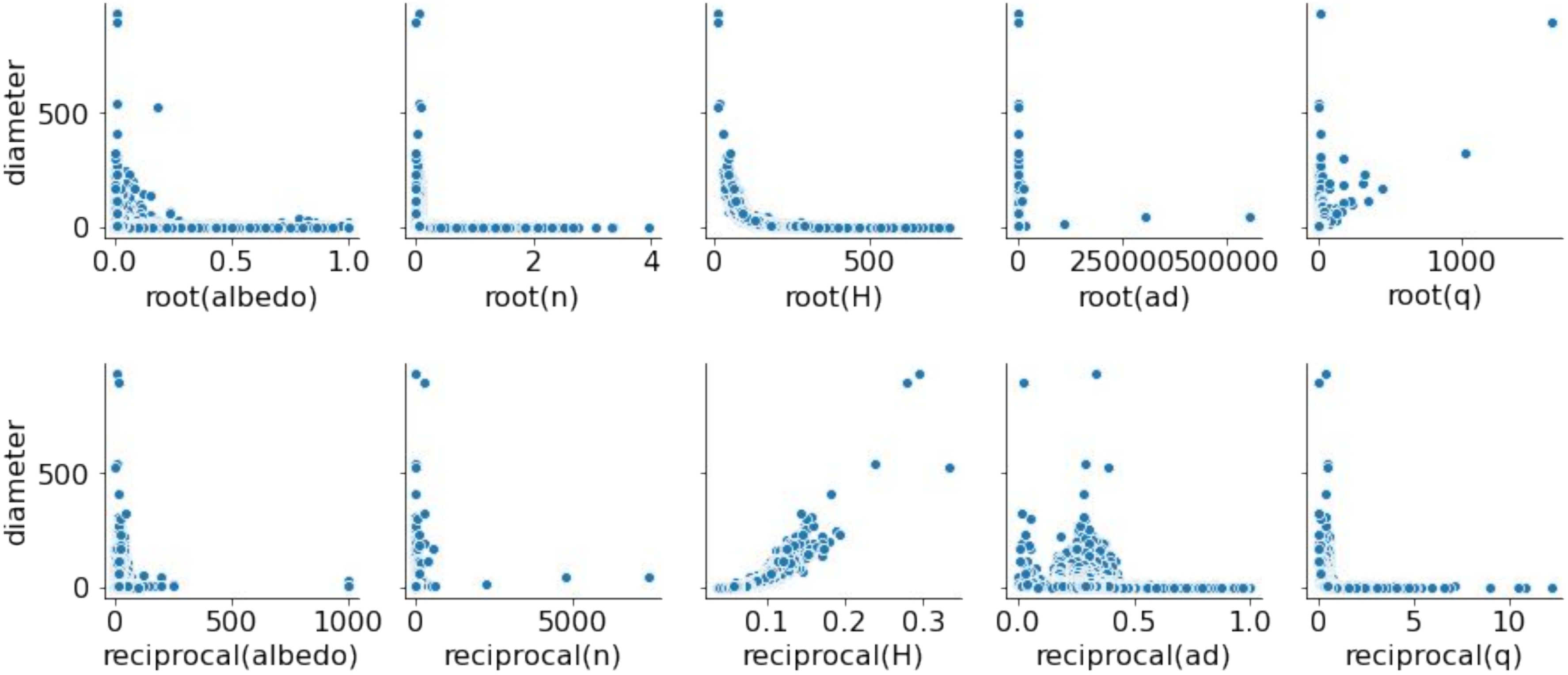


Analisi database

- Controlliamo anche le relazioni con eventuali trasformazioni...



Analisi database



Parte uno: Calcolo del diametro

Obiettivo: È possibile predire il valore corretto del diametro?

Modello: Linear, Ensemble, SVM, MLP, Tree e Nearest Neighbor

Misure di valutazione

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

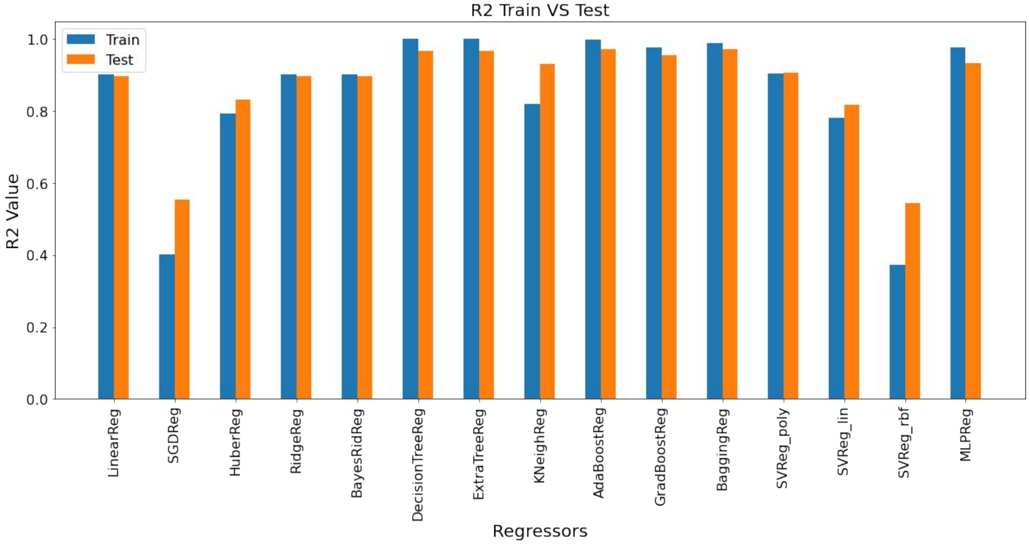
$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2$$

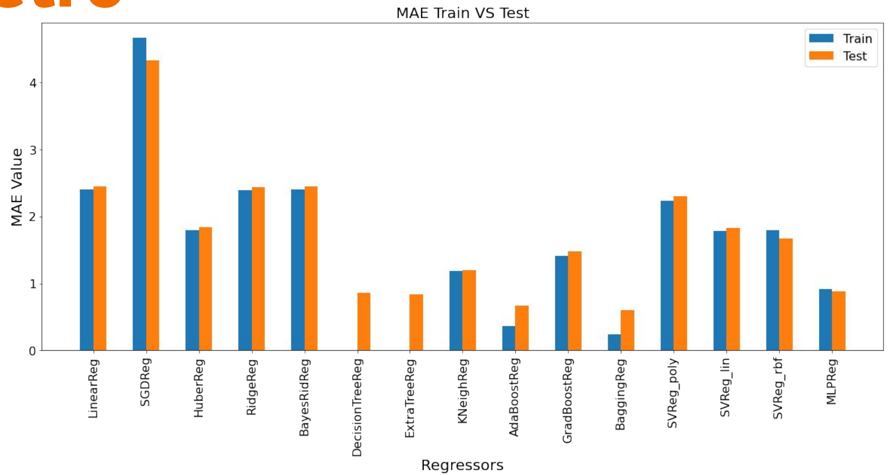
$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

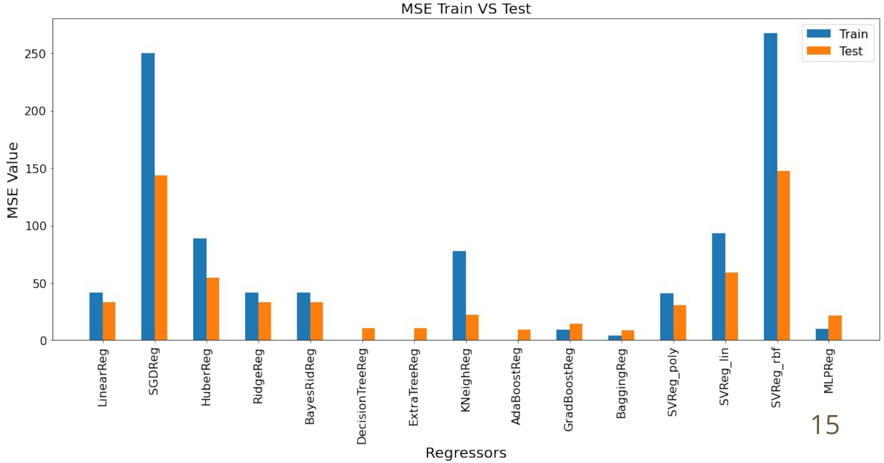
$\hat{y}$ – *predicted value of y*
$\bar{y}$ – *mean value of y*

Parte uno: Calcolo del diametro

Questo il comportamento dei regressori usati...





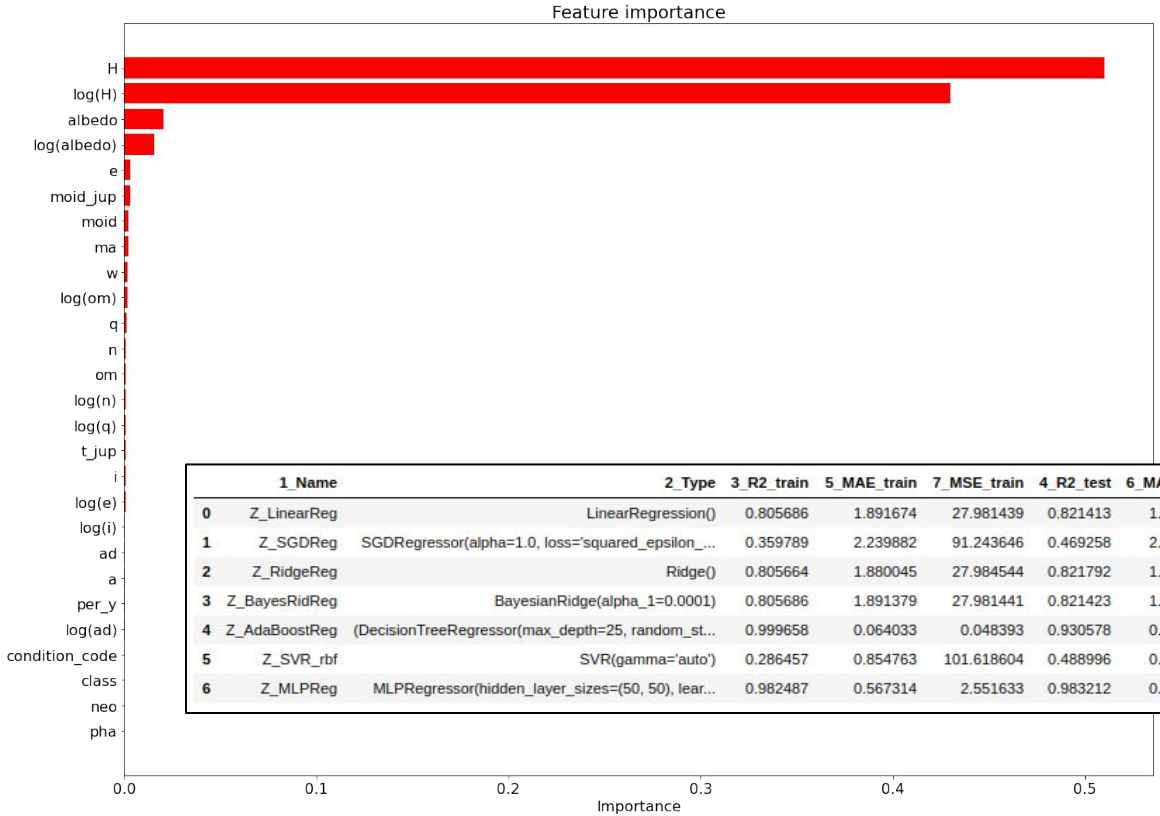


Bene modelli più complessi

Male modelli lineari e SVM

Parte uno: Calcolo del diametro

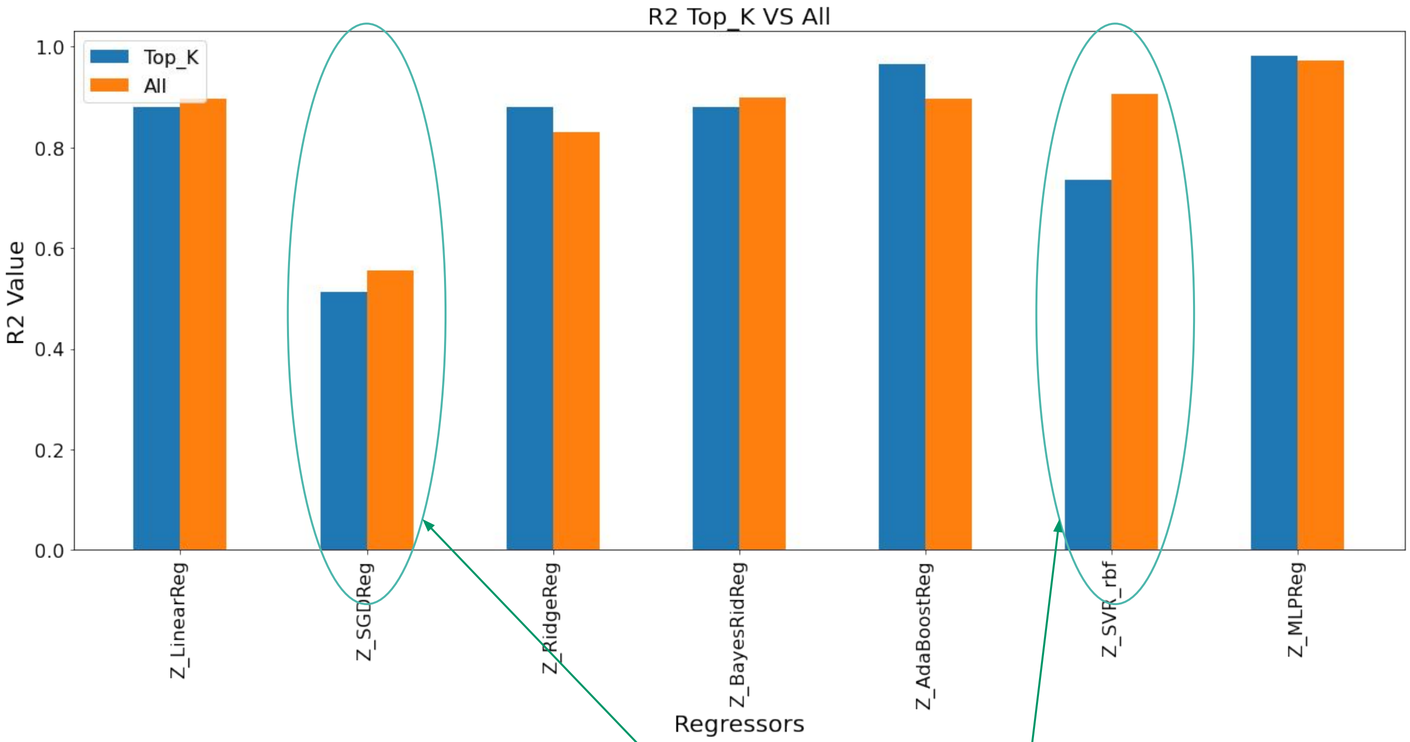
Possibile usare pochi ma buoni attributi?



	1_Name	2_Type	3_R2_train	5_MAE_train	7_MSE_train	4_R2_test	6_MAE_test	8_MSE_test
0	Z_LinearReg	LinearRegression()	0.805686	1.891674	27.981439	0.821413	1.777262	16.428951
1	Z_SGDReg	SGDRegressor(alpha=1.0, loss='squared_epsilon_...	0.359789	2.239882	91.243646	0.469258	2.960978	48.825249
2	Z_RidgeReg	Ridge()	0.805664	1.880045	27.984544	0.821792	1.768515	16.394148
3	Z_BayesRidReg	BayesianRidge(alpha_1=0.0001)	0.805686	1.891379	27.981441	0.821423	1.777036	16.428029
4	Z_AdaBoostReg	(DecisionTreeRegressor(max_depth=25, random_st...	0.999658	0.064033	0.048393	0.930578	0.561962	6.386463
5	Z_SVR_rbf	SVR(gamma='auto')	0.286457	0.854763	101.618604	0.488996	0.778283	47.009448
6	Z_MLPReg	MLPRegressor(hidden_layer_sizes=(50, 50), lear...	0.982487	0.567314	2.551633	0.983212	0.558491	1.544359

- Velocità training
- Modelli meno pesanti
- Misure di valutazione migliori?

Parte uno: Calcolo del diametro



Otteniamo una valutazione più o meno simile...

Rispetto all'utilizzo di tutte le features non migliora

Parte due: Classificazione fascia orbitale

Obiettivo: Predire la fascia di appartenenza di un corpo?

Modello: Linear, Tree, SVM, Nearest Neighbor, Naive Bayes e MLP

Misure di valutazione: Confusion matrix multiclass

Analizziamo Precision, Recall e F1-score nella versione *"macro"*

Simile a quella binaria ma estesa a più classi

Parte due: Classificazione fascia orbitale

Classi sbilanciate...

```
0     127090
1       7608
5       1876
7        740
2        612
4        555
3        337
8        125
6         52
10        12
9          7
Name: class, dtype: int64
```

- 0 MBA -> Main Belt Asteroid : Asteroids with orbital elements;
- 1 OMB -> Outer Main Belt : Asteroids with orbital elements;
- 2 MCA -> Mars Crossing Asteroid : Asteroids that cross the orbit of Mars;
- 3 AMO -> Amor : Near-Earth asteroid orbits similar to that of 1221 Amor;
- 4 IMB -> Inner Main Belt : Asteroids with orbital elements;
- 5 TJN -> Jupiter Trojan : Asteroids trapped in Jupiter's L4/L5 Lagrange points;
- 6 CEN -> Centaur : Objects with orbits between Jupiter and Neptune;
- 7 APO -> Apollo : Near-Earth asteroid orbits which cross the Earth's orbit similar to that of 1862 Apollo;
- 8 ATE -> Atene : Near-Earth asteroid orbits similar to that of 2062 Aten;
- 9 AST -> Asteroid : Asteroid orbit not matching any defined orbit class;
- 10 TNO -> TransNeptunian Object : Objects with orbits outside Neptune;
- 11 ETc -> Encke-type Comet : Encke-type comet;
- 12 COM -> Comete : Comet orbit not matching any defined orbit class;

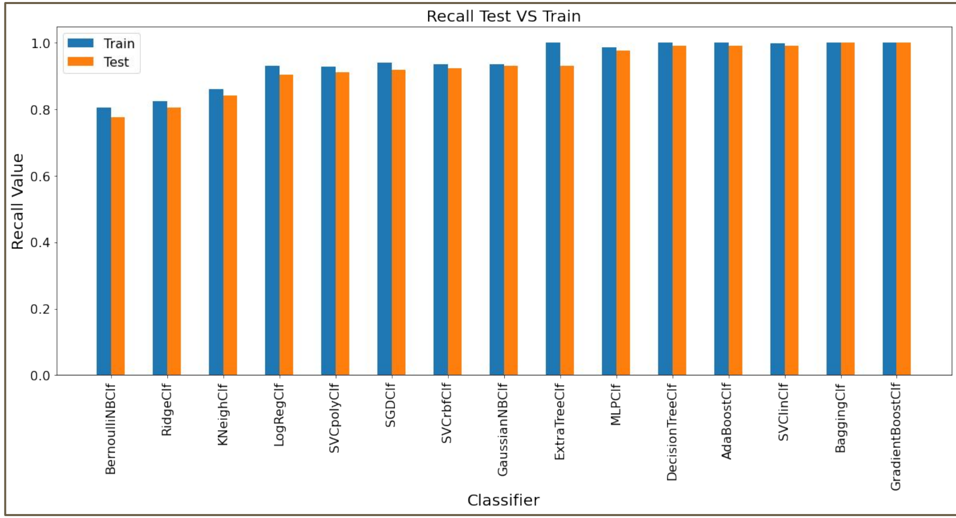
1. **Subsampling**
2. **Fusione classi (6,9,10 -> 6)**

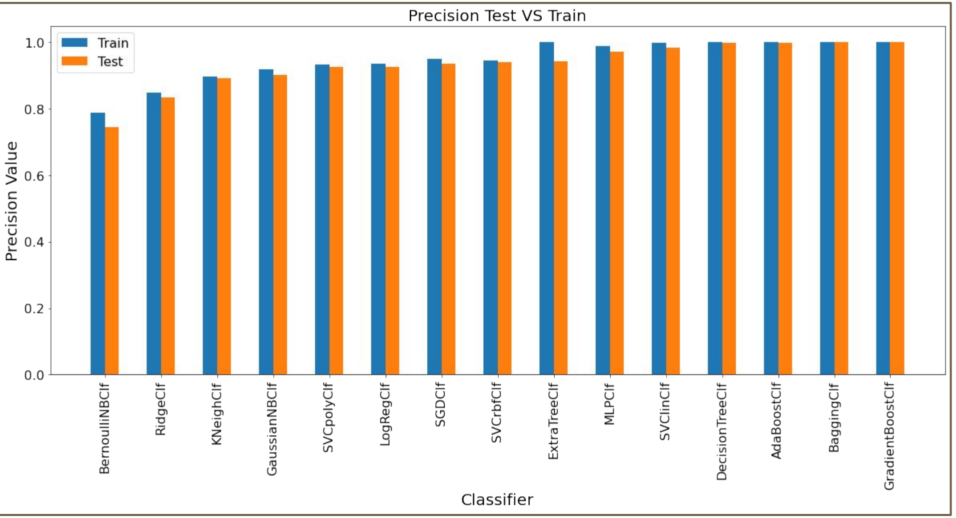
```
7     600
5     600
1     600
2     600
0     600
4     555
3     337
8     125
6      71
Name: class, dtype: int64
```

Parte due: Classificazione fascia orbitale

Vediamo subito ai risultati..

Accuratezza con classi sbilanciate?





Parte due: Classificazione fascia orbitale

Proviamo a migliorare con il Voting..

	1_name	2_model	3_accuracy_train	4_precision_train	5_recall_train	6_F1_train	3_accuracy_test	4_precision_test	5_recall_test	6_F1_test
0	Voting_hard_Clf	VotingClassifier(estimators=[('BernoulliNBClf'...	0.866514	0.879644	0.832940	0.846394	0.834963	0.857552	0.803476	0.820617
8	BernoulliNBClf	BernoulliNB()	0.811774	0.788766	0.804609	0.777243	0.773839	0.743740	0.777242	0.738420
0	RidgeClf	RidgeClassifier()	0.826529	0.848400	0.823742	0.827676	0.804401	0.833000	0.804384	0.808419

```
dt_copy = dt_clfs.copy()
dt_copy = dt_copy.loc[ dt_clfs['6_F1_test'] < 0.85 ]
print(dt_copy['1_name'].values)
```

```
['BernoulliNBClf' 'RidgeClf']
```

```
estimators = [ (x,y) for x,y in zip(dt_copy['1_name'], dt_copy['2_model'])]
ar = []

model = VotingClassifier(estimators=estimators, voting='hard')
model.fit(X_train, y_train)
clf = compute_metrics(model, 'Voting_hard_Clf', X_train, X_test, y_train, y_test)
```

Otteniamo un buon miglioramento lato accuratezza

L'aumento del F1-score legato a Precision e Recall

Parte tre: Anomaly Detection

Obiettivo: individuare i corpi PHA

Modello: Autoencoder (Feature extraction) + Logistic Regressor

Misure di valutazione: Confusion matrix

		Predetta	
		PHA	No PHA
Reale	PHA	TP	FN
	No PHA	FP	TN

$$Recall = \frac{TP}{TP + FN}$$

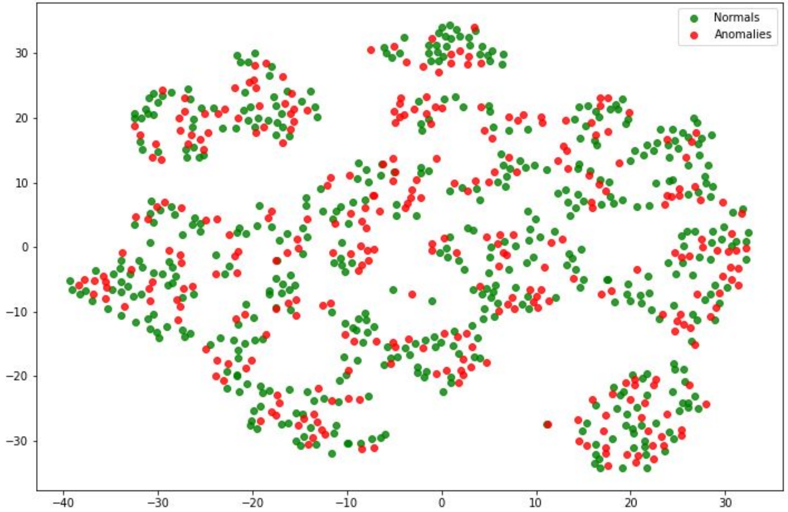
$$Precision = \frac{TP}{TP + FP}$$

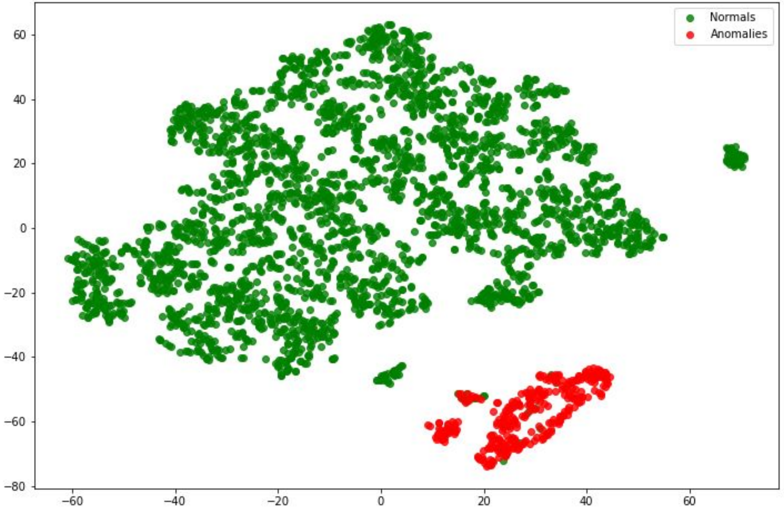
$$F1_{score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Parte tre: Anomaly Detection

1. Perché usare l'autoencoder?

2. Visualizzazione dei dati con t-SNE





Converte i valori di similarità in distribuzioni e minimizza la distanza di KL tra la probabilità ad alta dimensionalità e quella a bassa dimensionalità.

Parte tre: Anomaly Detection

E questo è il risultato..

Classification Report:

	precision	recall	f1-score	support
-1.0	0.892308	0.983051	0.935484	59.00000
1.0	0.998333	0.988449	0.993367	606.00000
accuracy	0.987970	0.987970	0.987970	0.98797
macro avg	0.945321	0.985750	0.964425	665.00000
weighted avg	0.988927	0.987970	0.988231	665.00000

Confusion matrix:

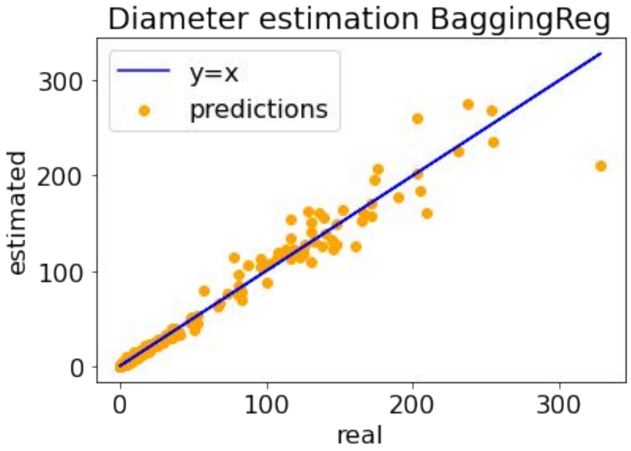
	0	1
0	58	1
1	7	599

Recall è importante alta!

Potremmo non osservare un PHA...

Prestiamo più attenzione

Conclusioni



La anomaly detection riesce nel riconoscimento dei PHA.

Questo task è particolarmente riuscito perché:

1. Recall e precision alti;
2. FN rate basso e possiamo analizzare pochi corpi;
3. L'autoencoder ci dà una "visione" diversa che aiuta nella distinzione;

Aiuta nel tracciare possibili missioni di PD (*Planetary Defense*).

La classificazione delle orbite risulta molto accurata.

- Presenza di buoni parametri descrittivi facilmente osservabili (magnitudo, albedo, velocità angolare);
- Anche nel peggiore dei casi, sono pochi i misclassificati (recall e precision alti);
- Record molto diversi con alcune feature (perielio, eccentricità, etc.).

I risultati delle misure sono buoni, tuttavia il problema si riscontra nei corpi il cui diametro è di più alta dimensione: l'errore commesso risulta essere più marcato. La probabile causa è da imputare alla mancanza di dati su corpi di maggiore dimensione.

La regressione del diametro, seppur buona, non è del tutto soddisfacente.

Appendice tecnica

- **Pandas :** gestione dei DataFrame e operazioni di preprocessing
- **Numpy :** gestione di valori in array, conversione dei tipi
- **Matplotlib** -> **pyplot :** gestire i grafici e le varie figure
- **Seaborn :** alternativa per la gestione dei grafici
- **Scikit-learn :** modellazione regressori, classificatori e metriche di valutazione
- **Keras :** costruzione dell'autoencoder

Pre-processing dei dati

- Gestione dati mancanti e *"null"*

Remove columns

```
#removing columns
vect = np.asarray(pd.DataFrame(data.isnull().sum() / data.shape[0]).T).flatten()
colstoRem = np.asarray(np.where(vect>0.5)).flatten().tolist()
print('index of columns to remove = ', colstoRem)
data.drop(data.columns[colstoRem], axis=1, inplace=True)
data.drop(['first_obs', 'n_obs_used', 'full_name'], axis=1, inplace=True)
data.info()
```

```
#remove meaningless rows
data = data[data.diameter.notnull()]
data = data[data.albedo.notnull()]
data = data[data.pha.notnull()]

data.info()
```

```
#filling
value = data.H.mode()[0]
data['H'].fillna(value, inplace=True)
```

- Trasformazione tipo dati

```
#converting into numerical
data['moid'] = pd.to_numeric(data['moid'])
data['pha'] = pd.to_numeric(data['pha'].map(dict(Y=1, N=0)))
data['neo'] = pd.to_numeric(data['neo'].map(dict(Y=1, N=0)))
data.info()
```

Pre-processing dei dati

- Trasformazione tipo dei dati

Convert class and condition code (or Uncertainty parameter U) into numerical attributes

```
classes = data['class'].unique()
codes = np.arange(classes.shape[0])
dict_classes = {}
for key,val in zip(classes, codes):
    dict_classes[key] = val

data['class'] = pd.to_numeric(data['class'].map(dict_classes))
data['condition_code'] = pd.to_numeric(data['condition_code'])
data.info()
```

- Trasformazione in scala logaritmica

```
for column in ['albedo','n','H','ad','e','om','i','q']:
    data['log('+column+')'] = data[column].apply(np.log)
data.head()
```

e	a	q	i	om	...	class	condition_code	log(albedo)	log(n)	log(H)	log(ad)	log(e)	log(om)	log(i)	log(q)
6009	2.769165	2.558684	10.594067	80.305531	...	0	0	-2.407946	-1.542316	1.223775	1.091805	-2.576903	4.385838	2.360294	0.939493
9972	2.773841	2.135935	34.832932	173.024741	...	0	0	-2.292635	-1.544847	1.435085	1.227225	-1.469797	5.153435	3.550563	0.758904
6936	2.668285	1.982706	12.991043	169.851482	...	0	0	-1.541779	-1.486651	1.673351	1.210113	-1.358927	5.134924	2.564260	0.684462
8721	2.361418	2.151909	7.141771	103.810804	...	0	0	-0.860856	-1.303390	1.098612	0.944266	-2.422253	4.642570	1.965961	0.766356
0913	2.574037	2.082619	5.367427	141.571026	...	0	0	-1.294627	-1.432710	1.931521	1.120196	-1.655936	4.952802	1.680349	0.733626

Regressori: grid-search e cross validation

```
def grid_searching(model, name, X_train, X_test, y_train, y_test, k_feature, params=None):
    clf = {}

    # grid searching
    if params is not None:
        grid_searcher = GridSearchCV(estimator=model, cv=5, n_jobs=-1, param_grid=params,
                                     scoring='neg_mean_squared_error')
        grid_searcher.fit(X_train, y_train)

        best_model = grid_searcher.best_estimator_
    else:
        best_model = model.fit(X_train, y_train)

    #cross_validating
    scores = cross_validate(best_model, X_train, y_train, cv=5, return_train_score=True,
                            scoring=['r2','neg_mean_absolute_error','neg_mean_squared_error'])
```

Miglior modello sulla base del parametro *"scoring"*

MSE e MAE per la regressione secondo scikit-learn

Regressori: modelli e risultati

TIPOLOGIA	REGRESSORE	G.SEARCH	PARAMETERES OF G.S.
Linear			
	Linear	✘	
	SGD	✔	alpha' : [1., 5., 10., 50., 100.], 'loss' : ['squared_loss', 'epsilon_insensitive', 'squared_epsilon_insensitive']
	Huber	✘	
	Ridge	✔	'alpha' : [1., 5., 10., 50., 100.]
	BayesianRidge	✔	alpha_1' : [0.0001, 0.001, 0.01, 0.1, 1., 5., 10., 50., 100.]
Tree			
	DecisionTree	✔	splitter': ['best', 'random'], 'criterion' : ['mse', 'friedman_mse', 'mae']
	ExtraTree	✔	splitter': ['best', 'random']
Nearest Neighbor			
	KNeigh	✔	n_neighbors' : [5, 10, 15, 50]
Ensemble			
	Adaboost	✔	n_estimators' : [5, 10, 25, 50], 'base_estimator' : [DecisionTreeRegressor(max_depth=3, 10, 15, 25, 30)
	Bagging	✔	n_estimators' : [5, 10, 25, 50], 'base_estimator' : [DecisionTreeRegressor(max_depth=3, 10, 15, 25, 30)
	GradientBoosting	✔	n_estimators' : [5, 10, 25, 30], 'loss' : ['ls','lad','quantile']
SVM			
	Linear	✔	C' : [0.1,1,10,50]
	Polynomial	✘	
	RBF	✘	
Neural Network			
	MLP	✔	hidden_layer_sizes' : [(10,), (50,), (100,), (10, 10,), (10, 50,), (10, 100), (50,10,), (50,50,), (50,100,), (100,10,), (100,50,), (100,100,)]

Classificatori: modelli e valutazioni

TIPOLOGIA	CLASSIFICATORE	PARAMETERES OF CLFs
Linear		
	Ridge	alpha=1.0
	SGD	alpha=0.0001, loss='hinge'
	LogisticRegressor	multi_class='ovr', solver'='lbfgs'
Tree		
	DecisionTree	criterion='gini', splitter='best'
	ExtraTree	criterion='gini', splitter='random'
Nearest Neighbor		
	KNeigh	n_neighbor=15, metric='minkowski', p=2
Ensemble		
	Adaboost	base_estimator=DecisionTree, n_estimator=50, algorithm='SAMME.R'
	Bagging	base_estimator=DecisionTree, n_estimator=50, criterion='gini'
	GradientBoosting	n_estimator=50, criterion='friedman_mse', learning_rate=0.1
SVM		
	Linear	C=100, decision_function_shape='ovr'
	Polynomial	degree=2
	RBF	decision_function_shape='ovr'
Neural Network		
	MLP	learning_rate='adaptive', hidden_layer_sizes=(10, 50,), activation='relu'
Naive Bayes		
	GaussianNB	var_smoothing=1e-09
	BernoulliNB	alpha=0.1

Classificatori: confusion matrix e cross validation

Adaboost

	0	1	2	3	4	5	6	7	8
0	124	0	0	0	0	0	0	0	0
1	0	126	0	0	0	0	0	0	0
2	0	0	118	0	0	0	0	0	0
3	0	0	0	72	0	0	0	0	0
4	0	0	0	0	110	0	0	0	0
5	0	0	0	0	0	118	0	0	0
6	0	0	1	0	0	0	10	0	0
7	0	0	0	0	0	0	0	111	0
8	0	0	0	0	0	0	0	0	28

Logistic Regr.

	0	1	2	3	4	5	6	7	8
0	82	37	4	0	1	0	0	0	0
1	7	119	0	0	0	0	0	0	0
2	0	0	110	0	8	0	0	0	0
3	0	0	0	66	0	0	0	6	0
4	0	0	0	0	110	0	0	0	0
5	0	0	0	0	0	118	0	0	0
6	0	0	1	0	0	1	9	0	0
7	0	0	0	7	0	0	0	104	0
8	0	0	0	0	0	0	0	1	27

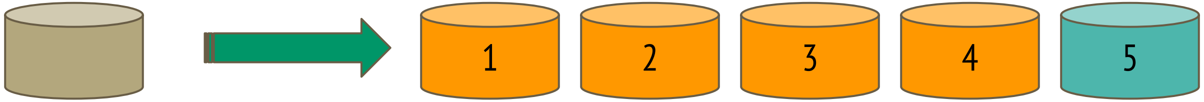
MLP

	0	1	2	3	4	5	6	7	8
0	118	4	2	0	0	0	0	0	0
1	5	120	0	0	0	0	1	0	0
2	0	0	115	0	3	0	0	0	0
3	0	0	0	68	0	0	0	4	0
4	0	0	0	0	110	0	0	0	0
5	0	0	0	0	0	118	0	0	0
6	0	0	0	0	0	0	11	0	0
7	0	0	0	2	0	0	0	109	0
8	0	0	0	0	0	0	0	1	27

BernoulliNB

	0	1	2	3	4	5	6	7	8
0	70	40	8	0	6	0	0	0	0
1	3	121	0	0	0	0	2	0	0
2	0	0	108	0	10	0	0	0	0
3	0	0	0	28	0	0	1	28	15
4	0	0	0	0	110	0	0	0	0
5	0	0	0	0	0	118	0	0	0
6	0	0	0	0	0	0	11	0	0
7	0	0	0	29	0	0	0	44	38
8	0	0	0	0	0	0	0	7	21

```python
scores = cross_validate(estimator=model, X=X_train, y=y_train, cv=5,
                        n_jobs=-1, scoring=['accuracy','precision_macro','recall_macro','f1_macro'],
                        return_train_score=True)
```



Autoencoder: modello e funzionamento

Impara a ricostruire tramite compressione e decompressione

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 27)]              0
_________________________________________________________________
dense (Dense)                (None, 20)                560
_________________________________________________________________
dense_1 (Dense)              (None, 10)                210
_________________________________________________________________
dense_2 (Dense)              (None, 20)                220
_________________________________________________________________
dense_3 (Dense)              (None, 27)                567
=================================================================
Total params: 1,557
Trainable params: 1,557
Non-trainable params: 0
_________________________________________________________________
```

```python
input_layer = Input(shape=(X.shape[1],))

## encoding part
encoded = Dense(20, activation='relu')(input_layer)
encoded = Dense(10, activation='relu')(encoded)

## decoding part
decoded = Dense(20, activation='relu')(encoded)

## output layer
output_layer = Dense(X.shape[1], activation='relu')(decoded)

autoencoder = Model(input_layer, output_layer)
autoencoder.compile(optimizer="adam", loss="mse")
autoencoder.summary()
```

Esegue feature extraction grazie alla rappresentazione latente al termine del *"bottleneck"*

Fine