Relazione

July 5, 2019

1 Progetto di Applicazioni Data Intensive 2018/2019

1.1 Descrizione del problema

Il problema da noi analizzato si pone l'obbiettivo di prevedere per conto di una società che effettua bike sharing (ossia una forma di affitto di biciclette automatizzato) il numero di biciclette che saranno affittate durante una giornata che presenta determinate caratteristiche.

Il dataset denominato "Bike Sharing Data Set" è stato scaricato dal sito https://archive.ics.uci.edu/ml e si compone di 731 istanze, 16 diversi attributi di cui 2 identificatori e 3 possibili soggetti di predizione.

[ ]:
```
import os.path
import csv
import pandas as pd
if os.path.exists("day.csv"):
    ds = pd.read_csv("day.csv", sep=",")
else:
    print("File non trovato")
```

[ ]:
```
len(ds.columns)
```

1.1.1 Descrizione variabili

instant: indice dei record. **dteday**: data. **season**: stagione **yr**: anno (0: 2011, 1: 2012) **mnth**: mese (1 - 12) **holiday**: giorno festivo (1: si, 0: no) **weekday**: giorno della settimana (0 - 6) **workingaday**: se il giorno è festivo o appartiene al weekend 0, altrimenti 1 **wheathersit**: condizioni meteo generali della giornata: * 1: soleggiato, poco nuvoloso * 2: nuvoloso, nebbia * 3: leggera neve, leggera pioggia * 4: neve, pioggia, fulmini

temp: temperatura media giornaliera (C) normalizzata. Valori divisi per 41. **atemp**: temperatura perepita (řC) normaizzata. Valori divisi per 50 **hum**: percentuale di umidità **windspeed**: velocità del vento normalizzata, Valori divisi per 67. **casual**: numero di utenti casuali **registered**: numero di utenti registrati **cnt**: numero di utenti totali

La variabile che abbiamo scelto per effettuare la predizione è la variabile "cnt" i quanto a fini di ricerca di mercato è la variabile che più interessa. escludendo dagli attributi le colonne "casual" e "registered" che di cui "cnt" è la somma.

[ ]:
```
ds.set_index(["dteday"], inplace=True)
dataset = ds.drop(["casual", "registered","instant"], axis=1)
Y = dataset["cnt"]
```

1.2 Analisi esplorativa

Una analisi dei vari attibuti si può avere attraverso la funzione *describe*

[ ]:
```
dataset.describe()
```

[ ]:
```
dataset.info(memory_usage="deep")
```

le seguanti funzioni consentono di calcolare la correlazione tra due serie e il grafico relativo:

[ ]:
```
import numpy as np
def getCorrelation(feature1, feature2):
    return np.mean((feature1-feature1.mean()) * (feature2-feature2.mean())) /␣
 ↪(feature1.std() * feature2.std())
```

[ ]:
```
import matplotlib.pyplot as plot
def plotData(x, y, XAxisName, YAxisName):
    plot.scatter(x, y)
    plot.grid()
    plot.xlabel(XAxisName); plot.ylabel(YAxisName)
    plot.show()
```

Mentre con la seguente si può ottenere una serie ordinata che indica la correlazione tra "cnt" e il nostro dataset indicizzato su "dtaday", un histogramma di ogni attributo, e un grafico a dispersione di ogni feature con l'obbiettivo

[ ]:
```
def correlationRank(dataset, feature):
    correlation = []
    for a in dataset.columns:
        correlation.append(getCorrelation(dataset[a].astype("float"), feature))
        plotData(dataset[a].astype("float"), feature, a, "Byke Rent")
    cor = pd.Series(correlation, dataset.columns)
    cor.sort_values(ascending=False, inplace=True)
    return cor
```

[ ]:
```
cor = correlationRank(dataset.drop(["cnt"], axis=1),dataset["cnt"])
```

[ ]:
```
print(cor)
```

Dai grafici e dal calcolo della correlazione scopriamo che gli attributi weekday workingday e holiday sono attributi poco importanti per il calcolo. Nonostante ciò essendo la correlazione un calcolo su un coefficente di primo grado questi attributi poco correlati non vengono esclusi dal calcolo in quanto questo si baserà verosimilmente su un algoritmo polinomiale.

1.3 Preaparazione dei dati

Molti dei dati sono già stati standardizzati alla creazione del dataset, i dati che normalmente vengono presentati come categorici sono già forniti in forma numerica

I dati che necessitano di standardidazione verranno elaborati successivamente in ogni Pipeline attraverso la funzione di sklearn *StandardScaler*

essendo questo un problema di regressione calcoliamo con la norma L1 le feature più rilevanti ma prima dividiamo il dataset in trainSet e ValidationSet

[ ]:
```
from sklearn.model_selection import train_test_split
Y = dataset["cnt"]
X = dataset.drop(["cnt"], axis=1)
XTrain, XVal, YTrain, YVal = train_test_split(X, Y, test_size=0.33,␣
 ↪random_state=73)
```

[ ]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import Lasso

def elaborationWithLasso(degeePipe=1, alphaPipe=0):
    return Pipeline([("poly", PolynomialFeatures(degree=degeePipe,␣
 ↪include_bias=False)),
                     ("scale",  StandardScaler()),
                     ("linreg", Lasso(alpha=alphaPipe, max_iter=6000, tol=0.
 ↪005))])
```

[ ]:
```
def showZerosFeatures(XTrain, YTrain):
    model = elaborationWithLasso(1, 2)
    model.fit(XTrain, YTrain)
    tmp = pd.Series(model.named_steps["linreg"].coef_, XTrain.columns)
    print(tmp)
    a = []
    for row in tmp.index:
        if(tmp[row]==0):
            a.append(row)
    print(a)
```

[ ]:
```
showZerosFeatures(XTrain, YTrain)
```

Prima di decidere se mantenere o no la colonna "temp" valutiamo la bontà del modello attraverso il calcolo del coefficiente R^2, dell' errore relativo e dell'errore quadratico medio.

[ ]:
```
def relativeError(YTrue, YPred):
    return np.mean(np.abs((YTrue - YPred) / YTrue))
```

[ ]:
```
from sklearn.metrics import mean_squared_error
def printEvalutation(X, Y, model):
    print("Mean squared error    : {:.5}".format(mean_squared_error(model.
 ↪predict(X), Y)))
    print("Relative error        : {:.5%}".format(relativeError(model.
 ↪predict(X), Y)))
    print("R-squared coefficient : {:.5}".format(model.score(X, Y)))
```

[ ]:
```
model = elaborationWithLasso(3, 8)
model.fit(XTrain, YTrain)
printEvalutation(XVal, YVal, model)
```

Il risultato è buono ma sono possibili miglioramenti per cui non andremo ad escludere manualmente gli attributi che la norma L1 azzera ma piuttosto riuseremo la norma successivamente.

1.4 Generazione modelli di learning

Generiamo adesso diversi modelli di learning utilizzando k (nested) cross fold validation applicata ad una grid search.

Definiamo innanzitutto le pipeline di ogni modello.

Il modello Ridge è un modello di regressione lineare che applica $ $\|\theta\|$

Valutiamo ora i modelli ricavati nel punto precedente attraverso le metriche gia introdotte di R^2, errore relativo e errore quadratico medio. aggiungiamo alla valutazione una tabella ottenuta attraverso l'attributo *cv_results* di *GridSearchCV* in modo da verificare quali parametri hanno portato un risultato migliore nei vari tipi di regressione.

[ ]:
```
def EvalutationTable(results):
    return pd.DataFrame(results.cv_results_).sort_values("mean_test_score",␣
 ↪ascending=False)
```

[ ]:
```
printEvalutation(XVal, YVal, lassoGridSearch)
EvalutationTable(lassoGridSearch)
```

[ ]:
```
printEvalutation(XVal, YVal, ridgeGridSearch)
EvalutationTable(ridgeGridSearch)
```

[ ]:
```
printEvalutation(XVal, YVal, NRGridSearch)
EvalutationTable(NRGridSearch)
```

[ ]:
```
printEvalutation(XVal, YVal, gs)
EvalutationTable(gs)
```

definiamo come modelli migliori i seguenti: * regressione con lasso di grado 6 e con $\lambda = 8$ * regressione con lasso di grado 6 e con $\lambda = 5$ * regressione con Elastic net di grado 6 con $\lambda = 2$ e $\alpha = 0.5$

che presentano rispettivamente: .....

[ ]:
```
LassoModel1 = Pipeline([("poly", PolynomialFeatures(degree=6,␣
 ↪include_bias=False)),
                        ("scale",  StandardScaler()),
                        ("linreg", Lasso(alpha=8, max_iter=6000, tol=0.005))])

LassoModel2 = Pipeline([("poly", PolynomialFeatures(degree=6,␣
 ↪include_bias=False)),
                        ("scale",  StandardScaler()),
                        ("linreg", Lasso(alpha=5, max_iter=6000, tol=0.005))])

ENModel = Pipeline([("poly",   PolynomialFeatures(degree = 6,␣
 ↪include_bias=False)),
                    ("scale",  StandardScaler()),
                    ("linreg",  ElasticNet(alpha=2, l1_ratio=0.5, tol = 0.05,␣
 ↪max_iter = 6000))])
```

[ ]:
```

```