Machine Learning with Python

by Ankit Rathi

AGENDA

- Breaking the ice
- Machine Learning warm-up
- Python warm-up
- Case Studies

Breaking the ice

- Who am I?
- Your background
- Setting the expectations

Who am I?

Analytics & Beyond

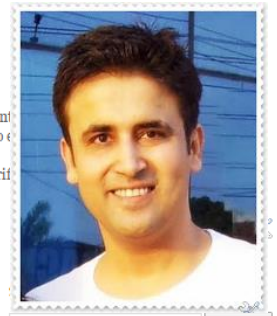
Welcome to a blog by just another Technology Enthusiast.

Since 2007, I am writing this blog to organize my learning on DB/DWH/ETL/BI/ML/Analytics/Big Data (Hadoop/Spark) concepts, architecture, development
posting/documenting my learning in my own way (notes, examples, workshops etc), but I try to mention the source of learning in the references for visitors to

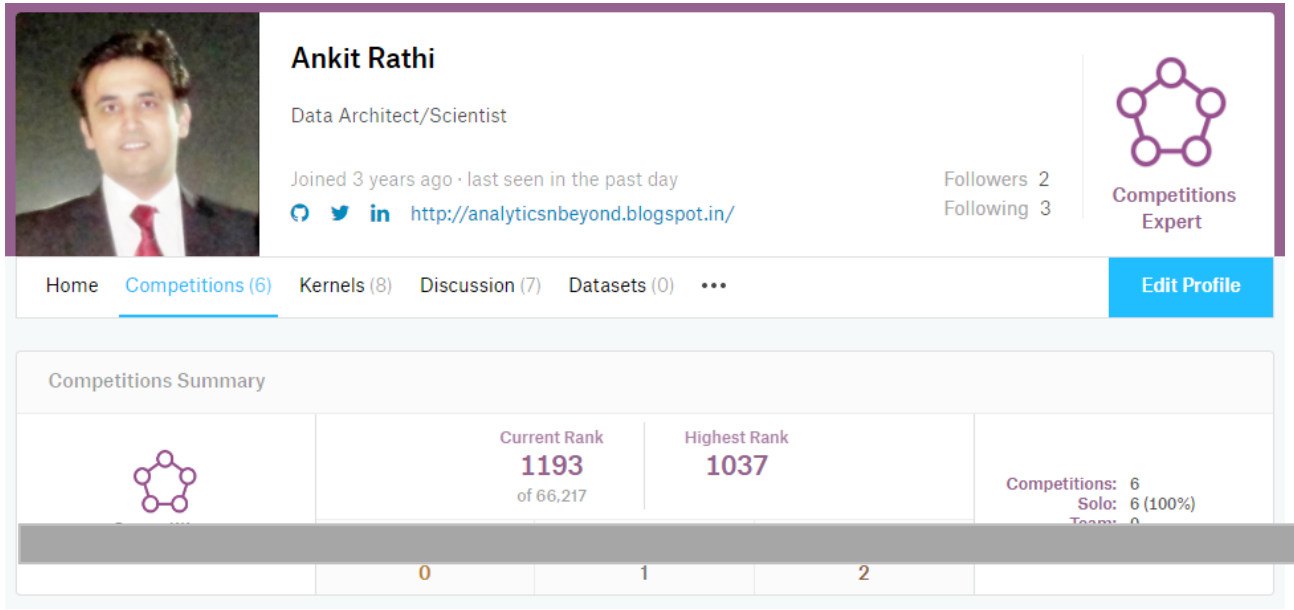
Again, this blog, not necessarily, covers the topic from scratch neither it promises to be accurate enough to be implemented directly, hence, please test and veri
environment before implementing.

I invite you for suggestions/feedback/queries/doubts.

Modern Data Architecture



Search



rathakt
ankitrathi169

What is your story?

- Class Intro

What's the deal?

- Setting the expectation

Machine Learning Warm-up

- Concepts
  - What? Why?
  - Supervised/Unsupervised/Reinforcement
- How?
  - Collect the data
  - Explore the data
  - Prepare the model
  - Evaluate the model
  - Tune the model
  - Deploy the model

Machine Learning Topics

- Linear Algebra (Matrices, Addition, Multiplication, Inverse, Transpose)
- Statistics (Descriptive & Inferential)
- Probability (Combinatorics/Conditional/Bayesian)
- Business Domain (Retail/Banking/Insurance/E-commerce)
- Data Related (DB/DWH/ETL/BI)

Python Warm-up

- Basics
  - Structures & Libraries
- How to:
  - Collect the data?
  - Explore the data?
  - Train the model?
  - Evaluate the model?
  - Tune the model?
  - Deploy the model?

Python Basics

- Structures
  - Lists, Arrays, Matrices & Dataframes
- Libraries
  - NumPy/SciPy
  - Pandas
  - matplotlib
  - scikit-learn

Data Collection

```
# import libraries
import pandas as pd
import json

# read csv file
df = pd.read_csv(“path", sep=‘seperator', encoding=‘encoding')

# read xls file
df= pd.read_excel(“path", sheetname=‘sheet’)

# read json file
with open('data.json') as data_file:
df = json.load(data_file)
```

Collect data from DB

```
import pandas as pd
import sqlite3

con = sqlite3.connect("data/portal_mammals.sqlite")

# Load the data into a DataFrame
surveys_df = pd.read_sql_query("SELECT * from surveys", con)

# Select only data for 2002
surveys2002 = surveys_df[surveys_df.year == 2002]

# Write the new DataFrame to a new SQLite table
surveys2002.to_sql("surveys2002", con, if_exists="replace")

con.close()
```

Data Merging

```
# append data row-wise
frames = [df1, df2, df3]
df= pd.concat(frames)

# OR
df = df1.append(df2)

# append data column-wise
df= pd.concat([df1, df2], axis=1)

# join data
df = pd.merge(df1, df2, on='key', how='outer')
```

Data Exploration I

```
# descriptive statistics summary
df['col'].describe()
# histogram
sns.distplot(df['col']);
# skewness and kurtosis
print("Skewness: %f" % df['col'].skew())
print("Kurtosis: %f" % df['col'].kurt())
# scatter plot col1/col2
var = 'col1'
data = pd.concat([df['col2'], df[var]], axis=1)
data.plot.scatter(x=var, y='col2', ylim=(0,800000));
```

Data Exploration II

```
# box plot col1/col2
var = 'col1'
data = pd.concat([df['col2'], df[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="col2", data=data)
fig.axis(ymin=0, ymax=800000);

# correlation matrix
corrmat = df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```

Feature Engineering I

```
# Imputation of Missing Data
from sklearn.preprocessing import Imputer
imp = Imputer(strategy='mean')
X1 = imp.fit_transform(X)
```

```
# Derived Features
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=3, include_bias=False)
X1 = poly.fit_transform(X)
```

Feature Engineering II

```
# Importing LabelEncoder and initializing it
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
# Iterating over all the common columns in train and test
for col in X_test.columns.values:
    # Encoding only categorical variables
    if X_test[col].dtypes=='object':
    # Using whole data to form an exhaustive list of levels
        data=X_train[col].append(X_test[col])
        le.fit(data.values)
        X_train[col]=le.transform(X_train[col])
```

Model Building

```
# define models
clf= RandomForestClassifier()
lr= LinearRegression()

# train the model
*mod.fit(X_train, y_train)

# predict on test data
y_pred = *mod.predict(X_test)

*mod ->  clf or lr
```

Model Evaluation

```
# Accuracy
accuracy = accuracy_score(y_test, y_pred)


#  K-fold cross-validation
scores = cross_val_score(clf, X_train, y_train, cv=5)
```

Model Tuning

```
# Hyper-parameter tuning
model = RandomForestRegressor(n_estimator = 200, oob_score = TRUE, n_jobs = -1,random_state
=50, max_features = "auto", min_samples_leaf = leaf_size)


# parameters to tune
parameter_candidates = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]


# Applying Grid Search CV
clf = GridSearchCV(estimator=svm.SVC(), param_grid=parameter_candidates, n_jobs=-1)
```

Model Deployment

```
import dill as pickle
filename = 'model_v1.pk'
# save the model
with open('../flask_api/models/'+filename, 'wb') as file:
        pickle.dump(grid, file)
# load the model
with open('../flask_api/models/'+filename ,'rb') as f:
    loaded_model = pickle.load(f)
# use the model
loaded_model.predict(test_df)
```

Case Studies

- Classification: https://www.kaggle.com/c/titanic

Tutorial: https://github.com/ankitrathi169/ItMLwP/blob/master/TitanicTutorial.ipynb

- Regression: https://www.kaggle.com/c/house-prices-advanced-regression-techniques

EDA: https://github.com/ankitrathi169/ItMLwP/blob/master/HousePricesEDA.ipynb

- Text Mining: https://www.kaggle.com/c/word2vec-nlp-tutorial

- Image Classification: https://www.kaggle.com/c/digit-recognizer