Data Science Questions & Answers for Interview practice

If you're looking for Data Science Interview Questions & Answers for Experienced or Fresher, you are at right place. There are lot of opportunities from many reputed companies in the world. According to research Data Science Market Expected to Reach $128.21 Billion With 36.5% CAGR Forecast To 2022. So, You still have opportunity to move ahead in your career in Data Science Analytics. Ativitti AI offers Advanced Data Science Interview Questions 2018 that helps you in cracking your interview & acquire dream career as Data Science Analyst

Q. What Is Data Science?

Data Science is a new area of specialization being developed by the Department of Mathematics and Statistics at Bowling Green State University. This field integrates math, statistics, and computer science to prepare for the rapidly expanding need for data scientists. Students seeking to pursue studies in Data Science should declare a mathematics major as entering freshmen, in anticipation of completing the specialization in years three and four.

Q. What are there quirements of a Data Science program?

The Data Science specialization requires three semesters of calculus (MATH 1310, MATH 2320, and MATH 2330 or MATH 2350), linear algebra (MATH 3320), introduction to programming (CS 2010), probability and statistics I (MATH 4410), and regression analysis (STAT 4020). In addition, the requirements include:

a) MATH 2950 Introduction to Data Science. This one-hour seminar would introduce freshmen students to a variety of data-science applications and give them an introduction to programming.

b) MATH 3430 Computing with Data. This course will focus on the data wrangling and data exploration computational skills in the context of a modern computing language such as Python or R.

c) MATH 3440 Statistical Programming. This course will focus on writing scripts and functions using a modern statistical language such as R.

d) MATH 4440 Statistical Learning. This course deals with modern methods for modeling data including a variety of supervised and unsupervised methods.

In addition, the student will be asked to choose two of the following seven courses:

a) MATH 4320 Linear Algebra with Applications

b) MATH 4420 Probability and Statistics II

c) MATH 4470 Exploratory Data Analysis

d) CS 2020 Object-Oriented Programming

e) STAT 4440 Data Mining in Business Analytics

f) CS 4400 Optimization Techniques

g) CS 4620 Database Management Systems

Q. Compare SAS, R, Python, Perl?

a) SAS is a commercial software. It is expensive and still beyond reach for most of the professionals (in individual capacity). However, it holds the highest market share in Private Organizations. So, until and unless you are in an Organization which has invested in SAS, it might be difficult to access one. R & Python, on the other hand are free and can be downloaded by any one.

b) SAS is easy to learn and provides easy option (PROC SQL) for people who already know SQL. Even otherwise, it has a good stable GUI interface in its repository. In terms of resources, there are tutorials available on websites of various university and SAS has a comprehensive

documentation. There are certifications from SAS training institutes, but they again come at a cost. R has the steepest learning curve among the 3 languages listed here. It requires you to learn and understand coding. R is a low level programming language and hence simple procedures can take longer codes. Python is known for its simplicity in programming world. This remains true for data analysis as well.

c) SAS has decent functional graphical capabilities. However, it is just functional. Any customization on plots are difficult and requires you to understand intricacies of SAS Graph package. R has the most advanced graphical capabilities among the three. There are numerous packages which provide you advanced graphical capabilities. Python capabilities will lie somewhere in between, with options to use native libraries (matplotlib) or derived libraries (allowing calling R functions).

d) All 3 ecosystems have all the basic and most needed functions available. This feature only matters if you are working on latest technologies and algorithms. Due to their open nature, R & Python get latest features quickly (R more so compared to Python). SAS, on the other hand updates its capabilities in new version roll-outs. Since R has been used widely in academics in past, development of new techniques is fast.

Q. Mention features of Teradata?

a) Parallel architecture: The Teradata Database provides exceptional performance using parallelism to achieve a single answer faster than a non-parallel system. Parallelism uses multiple processors working together to accomplish a task quickly.

b) Single datastore: The Teradata Database acts as a single data store, instead of replicating database for different purposes with the teradata database we can store the data once and use it for all applications. The Teradata database provides same connectivity for all systems.

c) Scalability: Scalability is nothing but we can add components to the system, the performance increase as linear. Scalability enables the system to grow to support more users/data/queries/complexity of queries without experiencing performance degradation.

Q. What does a data scientist do?

A data scientist represents an evolution from the business or data analyst role. The formal training is similar, with a solid foundation typically in computer science and applications, modeling, statistics, analytics and math. What sets the data scientist apart is strong business acumen, coupled with the ability to communicate findings to both business and IT leaders in a

way that can influence how an organization approaches a business challenge. Good data scientists will not just address business problems, they will pick the right problems that have the most value to the organization. A traditional data analyst may look only at data from a single source – a CRM system, for example – a data scientist will most likely explore and examine data from multiple disparate sources. The data scientist will sift through all incoming data with the goal of discovering a previously hidden insight, which in turn can provide a competitive advantage or address a pressing business problem. A data scientist does not simply collect and report on data, but also looks at it from many angles, determines what it means, then recommends ways to apply the data.

Q. How do Data Scientists Code in R?

R is a popular open source programming environment for statistics and data mining. The good news is that it is easily integrated into ML Studio. I have a lot of friends using functional languages for machine learning, such as F#. It's pretty clear, however, that R is dominant in this space. Polls and surveys of data miners are showing R's popularity has increased substantially in recent years. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors. R is a GNU project and is written primarily in C, Fortran.

Q. What is Machine Learning?

Machine learning represents the logical extension of simple data retrieval and storage. It is about developing building blocks that make computers learn and behave more intelligently. Machine learning makes it possible to mine historical data and make predictions about future trends. Search engine results, online recommendations, ad targeting, fraud detection, and spam filtering are all examples of what is possible with machine learning. Machine learning is about making data-driven decisions. While instinct might be important, it is difficult to beat empirical data.

Q. What is the use of Machine Learning?

Machine Learning is found in things we use every day such as Internet search engines, email and online music and book recommendation systems. Credit card companies use machine

learning to protect against fraud.
Using adaptive technology, computers recognize patterns and anticipate actions. Machine Learning is used in more complex applications such as:
1. Self-parking cars
2. Guiding robots
3. Airplane navigation systems (manned and unmanned),
4. Space exploration
5. Medicine

Q. What is Machine Learning best suited for?

Machine Learning is good at replacing labor-intensive decision-making systems that are predicated on hand-coded decision rules or manual analysis. Six types of analysis that Marchine Learning is well suited for are:
1. classification (predicting the class/group membership of items)
2. regression (predicting real-valued attributes)
3. clustering (finding natural groupings in data)
4. multi-label classification (tagging items with labels)
5. recommendation engines (connecting users to items)

Q. Define Boxplot?

In descriptive statistics, a boxplot, also known as a box-and-whisker diagram or plot, is a convenient way of graphically depicting groups of numerical data through their FIVE-NUMBER SUMMARIES (the smallest observation, lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation). A boxplot may also indicate which observations, if any, might be considered outliers.

Q. What are the most important machine learning techniques?

In **Associative rule learning** computers are presented with a large set of observations, all being made up of multiple variables. The task is then to learn relations between variables such us A & B 3C (if A and B happen, then C will also happen).
In **Clustering** computers learn how to partition observations in various subsets, so that each

partition will be made up of similar observations according to some well-defined metric. Algorithms like K-Means and DBSCAN belong also to this class.
In **Density estimation** computers learn how to find statistical values that describe data. Algorithms like Expectation Maximization belong also to this class.

Q. Why is it important to have a robust set of metrics for machine learning?

Any machine learning technique should be evaluated by using metrics for analytically assessing the quality of results. For instance: if we need to categorize objects such as people, movies or songs into different classes, precision and recall might be suitable metrics.
Precision is the ratio tp / tp+fp where tP is the number of true positives and fP is the number of false positives. **Recall** is the ratio tp / tP+fn where tP is the number of true positives and fn is the number of false negatives. True and false are attributes derived by using manually created data. Precision and Recall are typically reported in a 2-d graph known as P/R Curves, where different algorithmic graphs can be compared by reporting the achieved Precision for fixed values of Recall.
In addition F1 is another frequently used metric, which combines precision and Recall into a single value:
F1 = 2*precision*recall / precision+recall
Scikit-learn provides a comprehensive set of metrics for classification, clustering, regression, ranking and pairwise judgment'. As an example the code below computes Precision and Recall.
Code

```
import numpy as np
from skleam.metrics import precision_recall_curve
y_true = np.array([0,1,1,0, 1])
y_scores = np.array([0.5, 0.6, 0.38, 0.9, 1])
precision, recall, thresholds = precision_recall_curve(y_true, y_scores)
print precision
print recall
```

Q. Why are Features extraction and engineering so important in machine learning?

The Features are the selected variables for making predictions. For instance, suppose you'd like to forecast, if tomorrow there will be a sunny day, then you will probably pick features like humidity (a numerical value), speed of wind (another numeric value), some historical information

(what happened during the last few years), whether or not it is sunny today (a categorical value yes/no) and a few other features. Your choice can dramatically impact on your model for the same algorithm and you need to run multiple experiments in order to find what the right amount of data and what the right features are in order to forecast with minimal error. It is not unusual to have problems represented by thousands of features and combinations of them and a good feature engineer will use tools for stack ranking features according to their contribution in reducing the error for prediction.

Different authors use different names for different features including attributes, variables and predictors. In this book we consistently use features.

Features can be categorical such as marital status, gender, state of residence, place of birth, or numerical such as age, income, height and weight. This distinction is important because certain algorithms such as linear regression work only with numerical attributes and if categorical features are present, they need to be somehow encoded into numerical values.

In other words, feature engineering is the art of extracting, selecting and transforming essential characteristics representing data. It is sometimes considered less glamourous than machine learning algorithms but in reality any experienced Data Scientist knows that a simple algorithm on a well-chosen set of features performs better than a sophisticated algorithm on a not so good set of features.

Also simple algorithms are frequently easier to implement in a distributed way and therefore they scale well with large datasets. So the rule of thumb is in what Galileo already said many centuries ago: “Simplicity is the ultimate sophistication”. Pick your algorithm carefully and spend a lot of time in investigating your data and in creating meaningful summaries with appropriate feature engineering.

Real world objects are complex and features are used to analytically represent those objects. From one hand this representation has an inherent error which can be reduced by carefully selecting a right set of representatives. From the other hand we might not want to create a too complex representation because it might be computationally expensive for the machine to learn a sophisticate model, indeed such model could possibly not generalize well to the unseen data.

Real world data is noisy. We might have very few instances (outliers) which show a sensible difference from the majority of the remaining data, while the selected algorithm should be resilient enough to outliers.

Real world data might have redundant information. When we extract features, we might be interested in optimizing simplicity of learned models and discard new features which show a high correlation with the already observed ones.

ETL is the process of Extraction, Transformation and Loading of features from real data for creating various learning sets. Transformation in particular refers to operations such as features weighting, high correlated features discarding, the creation of synthetic features derivative of the

one observed in the data and the reduction of high dimension features space into a lower one by using either hashing or rather sophisticate space projection techniques. For example in this book we discuss:

1. TFxIDF, an example of features weighting used in text classification
2. ChiSquare, an example of filtering of highly correlated features
3. Kernel Trick, an example of creation of derivative features
4. Hashing, a simple technique to reduce feature space dimensions
5. Binning, an example of transformation of continuous features into a discrete one. New synthetic features might be created in order to represent the bins.

Q. What is a Bias – Variance tradeoff?

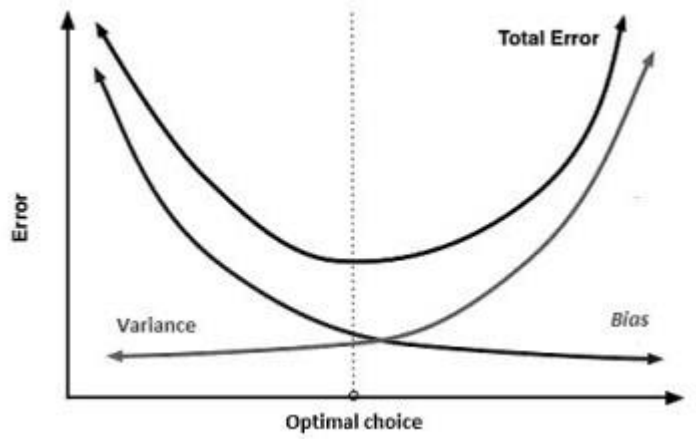
Bias and Variance are two independent sources of errors for machine learning which prevent algorithms to generalize the models learned beyond the training set.

a) Bias is the error representing missing relations between features and outputs. In machine learning this phenomenon is called underfitting.

b) Variance is the error representing sensitiveness to small training data fluctuations. In machine learning this phenomenon is called overfitting.

A good learning algorithm should capture patterns in the training data (low bias), but it should also generalize well with unseen application data. In general a complex model can show low bias because it captures many relations in the training data and, at the same time, it can show high variance because it will not necessarily generalize well. The opposite happens with models with high bias and low variance. In many algorithms an error can be analytically decomposed in three components: bias, variance and the irreducible error representing a lower bound on the expected error for unseen sample data.

One way to reduce the variance is to try to get more data or to decrease the complexity of a model. One way to reduce the bias is to add more features or to make the model more complex, as adding more data will not help in this case. Finding the right balance between Bias and Variance is an art that every Data scientist must be able to manage.



Q. What is a cross-validation and what is an overfitting?

Learning a model on a set of examples and testing it on the same set is a logical mistake because the model would have no errors on the test set but it will almost certainly have a poor performance on the real application data. This problem is called overfitting and it is the reason why the gold set is typically split into independent sets for training, validation and test. An example of random split is reported in the code section, where a toy dataset with diabetics' data has been randomly split into two parts: the training set and the test set.
As discussed in the previous question: given a family of learned models, the validation set is used for estimating the best hyper-parameters. However by adopting this strategy there is still the risk that the hyper-parameters overlit a particular validation set.
The solution to this problem is called cross-validation. The idea is simple: the test set is split in k smaller sets called folds and the model is then learned on k – 1 folds, while the remaining data is used for validation. This process is repeated in a loop and the metrics achieved for each iterations are averaged. An example of cross validation is reported in the section below where our toy dataset is classified via SVM and accuracy is computed via cross-validation. 5VM is a classification technique and "accuracy" is a quality measurement, which will be discussed later in the book.
Stratified KFold is a variation of k-fold where each set contains approximately the same balanced percentage of samples for each target class as the complete set.
Code

```
import numpy as np
front skleam import cross_validation
from skleam import datasets
from skleam import svm
```

diabets = datasets.load_diabetes()
X_train, X_test, y_train, y_test =
cross_validation.train_test_split(
diabets.data, diabets.target, test_size*).2, random_state=0)
print X_train.shape, y_trainshape # test size 20%
print X_test.shape, y_test.shape
cif = svm.SVC(kemel=qineari, C=1)
scores = cross_validation.cross_valscore(
clf, diabets.data, diabets.target, cv=4) # 4-folds
print scores
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.meanQ, scores.std()))

Q. Why are vectors and norms used in machine learning?

Objects such as movies, songs and documents are typically represented by means of vectors of features. Those features are a synthetic summary of the most salient and discriminative objects characteristics. Given a collection of vectors (the so-called vector space) V, a norm on V is a function P: V A satisfying the following properties: For all complex numbers a and all
u, v E V,
1. P(av) = |a| p(v)
2. P(u+v) <= p(u) + p(v)
3. If p(v) = 0 then v is the zero vector
The intuitive notion of length for a vector is captured by
||x||2 = root (x12 + … + xn2)
More generally we have

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

The special case $||x||_\infty$ is defined as

$$||x||_\infty = \max_i |x_i|$$

Code
from numpy import linalg as LA import numpy as np a = np.arange(22) print LA.nonn(a) print LA.nonn(a, I)

Q. What are Numpy, Scipy and Spark essential datatypes?

Numpy provides efficient support for memorizing vectors and matrices and for linear algebra operations 5. For instance: dot(a, b[, out]) is the dot product of two vectors, while inner(a, b) and outer(a, b[, out]) are respectively the inner and outer products.
Scipy provides support for sparse matrices and vectors with multiple memorization strategies in order to save space when dealing with zero entries.6 In particular the COOrdinate format specifies the non-zero v value for the coordinates(•), while the Compressed Sparse Colum matrix (CSC) satisfies the relationship M[row_ind[k], col_ind[k]] = data[k]
Spark has many native datatypes for local and distributed computations. The primary data abstraction is a distributed collection of items called “Resilient Distributed Dataset (RDD)”. RDDs can be created from Hadoop InputFormats’
HTTP://DOCS.SCIPY.ORGIDOCINUMPY/REFERENCE/ROUTINES.LINALG.HTML
HTTP://DOCS.SCIPY.ORGICLOCISCIPYIREFERENCE/SPARSE.HTML
or by transforming other RDDs. Numpy arrays, Python list and Scipy CSC sparse matrices are all supported. In addition: MLIB, the Spark library for machine learning, supports SparseVectors and LabeledPoint, i.e. local vectors, either dense or sparse, associated with a label/response
Code

```
import numpy as np
from scipy.sparse import csr_matrix
M = csr_matrix ([[4, 1, 0], [4, 0, 3], [0, 0, 1]])
from pyspark.mllib.linalg import SparseVector
from pyspark.mllib.regression import LabeledPoint
label = 0.0 point = LabeledPoint(label. SparseVector(3, [0, 2], [1.0, 3.0]))
textRDD = sc.textFile(“README.md”)
print textRDD.count() # count items in RDD
```

Q. Can you provide an example of features extraction?

Let’s suppose that we want to perform machine learning on textual files. The first step is to extract meaningful feature vectors from a text. A typical representation is the so called bag of words where:
1. Each word w in the text collection is associated with a unique integer = wordId(w) assigned to it.
2. For each document i, the number of occurrences of each word w is computed and this value is stored in a matrix M(i.D. Please, note that M is typically a sparse matrix because when a word is

not present in a document, its count will be zero.

Numpy, Scikit-learn and Spark all support sparse vectors2. Let's see an example where we start to load a dataset made up of Usenet articles; where the altatheism category is considered and the collection of text documents is converted into a matrix of token counts. We then print the wordId('man).

Code

From sklearn.datasets import fetch_20 new groups

HTTP://DOCS.SCIPY.ORG/DOC/SCIPY/REFERENCE/SPARSE.HTML

HTTP://SCIKIT-LEARN.ORG/STABLE/DATASETS/

Two additional observations can be here highlighted: first, the training set, the validation set and the test set are all sampled from the same gold set but those samples are independent. Second, it has been assumed that the learned model can be described by means of two different functions f and h combined by using a set of hyper-parameters A.

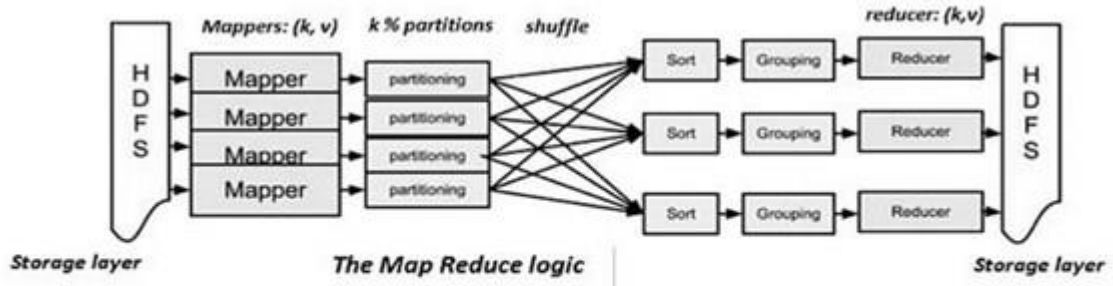
Unsupervised machine learning consists in tests and application phases only because there is no model to be learned a-priori. In fact unsupervised algorithms adapt dynamically to the observed data.

Q. Can you provide an example for Map and Reduce in Spark? (Let's compute the Mean Square Error)

Spark is a powerful paradigm for parallel computations which are mapped into multiple servers with no need of dealing with low level operations such as scheduling, data partitioning, communication and recovery. Those low level operations were typically exposed to the programmers by previous paradigms. Now Spark solves these problems on our behalf.

A simple form of parallel computation supported by Spark is the "Map and Reduce" which has been made popular by Google.

HTTP://RESEARCH.GOOGLE.COM/ARCHIVE/MAPREDUCE.HTML



Mappers receive (key, value) pairs and output (key, value)
Partitioners split the keys into partitions and then shuffle the data
Sorters perform the grouping
Reducers receive (key, iterable[value]) and output (key, value)

In this framework a set of keywords is mapped into a number of workers (e.g. parallel servers available for computation) and the results are then reduced (e.g. collected) by applying a “reduce” operator. The reduce operator could be very simple (for instance a sum) or sophisticated (e.g. a user defined function).
As an example of distributed computation let’s compute the Mean Square Error (MSE), the average of the squares of the difference between the estimator and what is estimated.
In the following example we suppose that valuesAndPreds is an RDD of many (vi = true labels, pi = predictions) tuples. Those are mapped into values (vi – pi)2. All intermediate results computed by parallel workers are then reduced by applying a sum operator. The final result is then divided by the total number of tuples as defined by the mathematical definition MSE = 1/n nEi=1 (vi – pi)2. Note that Spark hides all the low level details to the programmer by allowing to write a distributed code which is very close to a mathematical formulation.
Code
MSE = valuesAndPreds.rnap(lambda (v, p): (v – p)**2).reduce(lanthda x, y: x + y) / valuesAndPreds.count()
Spark can however support additional forms of parallel computation by taking inspiration from the 20 years of work on skeletons computations and, more recently, on Microsoft’s Cosmos.

Q. Can you provide examples for other computations in Spark?
The first code fragment is an example of map reduction, where we want to find the line with most words in a text. First each line is mapped into the number of words it contains. Then those numbers are reduced and the maximum is taken. Pretty simple: one single line of code stays here for something which requires hundreds of lines in other parallel paradigms such as Hadoop.
Spark supports two types of operations: transformations, which create a new RDD dataset from an existing one, and actions, which return a value to the driver program after running a computation on the dataset. All transformations in Spark are lazy because they postpone computation as much as possible until the results are really needed by the program. This allows Spark to run efficiently — for example the compiler can realize that an RDD created through map will be used in a reduce and return only the result of the reduce to the driver, rather than the larger mapped dataset. Intermediate results can be persisted and cached.
Basic transformations include (the list below is not comprehensive. Check online for a full list)

Transformation	**Use**
Map(func)	Returns a new distributed dataset formed by passing each element of the

	source through a function func.
filter(func)	Returns a new dataset formed by selecting those elements of the source on which func returns true.
flatMap(func)	Similar to map, but each input item can be mapped to 0 or more output items (so func should return a Seq rather than a single item).
sample(withReplacement, fraction, seed)	Samples a fraction of the data, with or without replacement, using a given random number generator seed.
union(otherDataset)	Returns a new dataset that contains the union of the elements in the source dataset and argument.
intersection(otherDataset)	Returns a new ROD that contains the intersection of elements in the source dataset and argument.
distinct([numTasks]))	Returns a new dataset that contains the distinct elements of the source dataset.
groupByKey([numTasks])	When called on a dataset of (K, V) pairs, a dataset of (K, Iterable) pairs returns.
reduceByKey(func, [numTasks])	When called on a dataset of (K, V) pairs, a dataset of (K, V) pairs returns, where the values for each key are aggregated using the given reduce function func, which must be of type (V,V) => V
sortByKeyflascending], [numTasks])	When called on a dataset of (K, V) pairs, where K implements Ordered, a dataset of (K, V) pairs sorted by keys in ascending or descending order returns, as specified in the Boolean ascending argument.