Q1. You are given a train data set having 1000 columns and 1 million rows. The data set is based on a classification problem. Your manager has asked you to reduce the dimension of this data so that model computation time can be reduced. Your machine has memory constraints. What would you do? (You are free to make practical assumptions).

Answer:

Processing a high dimensional data on a limited memory machine is a strenuous task, your interviewer would be fully aware of that. Following are the methods you can use to tackle such situation:

1. Since we have lower RAM, we should close all other applications in our machine, including the web browser, so that most of the memory can be put to use.
2. We can randomly sample the data set. This means, we can create a smaller data set, let's say, having 1000 variables and 300000 rows and do the computations.
3. To reduce dimensionality, we can separate the numerical and categorical variables and remove the correlated variables. For numerical variables, we'll use correlation. For categorical variables, we'll use chi-square test.
4. Also, we can use PCA and pick the components which can explain the maximum variance in the data set.
5. Using online learning algorithms like Vowpal Wabbit (available in Python) is a possible option.
6. Building a linear model using Stochastic Gradient Descent is also helpful.
7. We can also apply our business understanding to estimate which all predictors can impact the response variable. But, this is an intuitive approach, failing to identify useful predictors might result in significant loss of information**.**

Q2. You are given a data set. The data set has missing values which spread along 1 standard deviation from the median. What percentage of data would remain unaffected? Why?

Answer: let's assume it's a normal distribution. So, for a normal distribution there are approximately 68% of the values which lie within 1 standard deviation from the mean. similarly, there are 95 % of values which lie within 2 standard deviations of the mean and there are 99% of values which lie within 3 standard deviations from the mean. Assuming that the median is very close to the mean, we can say that approximately 32% of the values remain unaffected.

Q3. You are given a data set on cancer detection. You've build a classification model and achieved an accuracy of 96%. Why shouldn't you be happy with your model performance? What can you do about it?

Answer:

If you have worked on enough data sets, you should deduce that cancer detection results in imbalanced data. In an imbalanced data set, accuracy should not be used as a measure of performance because 96% (as given) might only be predicting majority class correctly, but our class of interest is minority class (4%) which is the people who actually got diagnosed with cancer. Hence, in order to evaluate model performance, we should use Sensitivity (True Positive Rate), Specificity (True Negative Rate), F measure to determine class wise performance of the classifier. If the minority class performance is found to to be poor, we can undertake the following steps:

1. We can use undersampling, oversampling or SMOTE to make the data balanced.
2. We can alter the prediction threshold value by doing probability caliberation and finding a optimal threshold using AUC-ROC curve.
3. We can assign weight to classes such that the minority classes gets larger weight.
4. We can also use anomaly detection.

Q4. Explain prior probability, likelihood, and marginal likelihood in the context of naïve Bayes algorithm.

Answer:

According to the Bayes theorem, the likelihood of a hypothesis (H) given evidence (E) is equal to the likelihood of the evidence given the hypothesis times the likelihood of the hypothesis itself. The following is how it is expressed mathematically −

P(H|E) = P(E|H) * P(H) / P(E)

where P(E) is the marginal likelihood, P(H|E) is the posterior probability of the hypothesis given the evidence, and P(E|H) is the likelihood of the evidence given the hypothesis.

Prior probability

The probability of each class before any characteristics are observed is known as the prior probability in the Naive Bayes method. The prior probability of class A, for instance, would be the likelihood that an item belongs to class A before witnessing any characteristics in a binary classification problem with classes A and B. The likelihood that an item belongs to class B before viewing any features would be known as the prior probability of class B.

Likelihood

In the Naive Bayes method, the likelihood is the likelihood of detecting each feature given the class. The likelihood of feature X1 given class A would be the chance of detecting feature X1 in objects belonging to class A, for instance, if there are two features, X1 and X2, and two classes, A and B. The chance of witnessing feature X2 in objects belonging to class B would be the likelihood of feature X2 given class B.

Marginal likelihood

The probability of witnessing the evidence is known as the marginal likelihood in the Naive Bayes method. The set of features that have been seen for an item is considered evidence in the Naive Bayes method. The evidence would be "X1, not X2," for instance, if there are two characteristics, X1 and X2, and an item possesses X1 but not X2.

Q5. You are working on a time series data set. You manager has asked you to build a high accuracy model. You start with the decision tree algorithm, since you know it works fairly well on all kinds of data. Later, you tried a time series regression model and got higher accuracy than decision tree model. Can this happen? Why?

Answer:

Time series data is known to posses linearity. On the other hand, a decision tree algorithm is known to work best to detect non – linear interactions.
yes, it is possible for a time series regression model to achieve higher accuracy than a decision tree model on a time series data set. This can happen for several reasons. First, time series regression models are specifically designed to handle time series data, which means that they are better able to capture the underlying patterns and relationships in the data. This can make them more accurate than other types of models, such as decision trees, which are not specifically designed for timeseries data.

Second, decision trees can be prone to overfitting on time series data, especially if the data is complex or noisy. Overfitting can lead to lower accuracy and poorer performance on unseen data. In contrast, time series regression models are less susceptible to overfitting, which can make them more accurate in these situations.

Q6. You are assigned a new project which involves helping a food delivery company save more money. The problem is, company's delivery team aren't able to deliver food on time. As a result, their customers get unhappy. And, to keep them happy, they end up delivering food for free. Which machine learning algorithm can save them?

Answer:

This is not a machine learning problem. This is a route optimization problem. A machine learning problem consist of three things:

1. There exist a pattern.
2. You cannot solve it mathematically (even by writing exponential equations).
3. You have data on it.

Addressing the issue of food deliveries not being on time involves optimizing the delivery process rather than relying solely on a machine learning algorithm. However, machine learning can contribute to improving delivery efficiency. Here are some ways machine learning an be applied:

1. Route Optimization:

- Use machine learning algorithms to analyze historical delivery data and predict optimal routes based on factors such as traffic patterns, time of day, and day of the week. This can reduce delivery times and improve overall efficiency.

2. Demand Prediction:

- Predicting the demand for food delivery at different times and locations can help the company allocate resources more effectively. Machine learning algorithms, such as time series forecasting or regression models, can be employed for demand prediction.

3.Dynamic Pricing:

- Implement machine learning algorithms for dynamic pricing to incentivize customers to choose delivery times with lower demand, thereby spreading out delivery requests more evenly throughout the day.

4. Predictive Maintenance:

- If the company uses a fleet of vehicles, implement predictive maintenance models to anticipate and address potential issues with delivery vehicles before they cause delays.

5. Real-time Monitoring:

- Use machine learning for real-time monitoring of delivery operations. This can involve tracking delivery progress, identifying potential bottlenecks, and dynamically adjusting routes based on current conditions.

6. Customer Behavior Analysis:

- Analyze customer behavior data to understand preferences, peak ordering times, and delivery location patterns. This information can be used to optimize delivery schedules and plan resources accordingly.

7. Optimal Staffing:

- Predict staffing needs based on historical and real-time data. Machine learning algorithms can help determine the optimal number of delivery personnel required at different times and locations.

Q7. You came to know that your model is suffering from low bias and high variance. Which algorithm should you use to tackle it? Why?

Answer:

Low bias occurs when the model's predicted values are near to actual values. In other words, the model becomes flexible enough to mimic the training data distribution. While it sounds like great achievement, but not to forget, a flexible model has no generalization capabilities. It means, when this model is tested on an unseen data, it gives disappointing results. In such situations, we can use bagging algorithm (like random forest) to tackle high variance problem. Bagging algorithms divides a data set into subsets made with repeated randomized sampling. Then, these samples are used to generate a set of models using a single learning algorithm. Later, the model predictions are combined using voting (classification) or averaging (regression).

Also, to combat high variance, we can:

1. Use regularization technique, where higher model coefficients get penalized, hence lowering model complexity.
2. Use top n features from variable importance chart. May be, with all the variable in the data set, the algorithm is having difficulty in finding the meaningful signal.

Q8. Problem based on different machine Learning algorithm. Take any data set and apply the ML algorithm such as Classification and Clustering .

Answer:

Let's consider a common dataset for this demonstration: the Iris dataset, which is often used for classification and clustering tasks. The Iris dataset contains measurements of sepal length, sepal width, petal length, and petal width for three species of iris flowers (setosa, versicolor, and virginica).

We'll use two machine learning algorithms:

Classification Algorithm: Random Forest
Clustering Algorithm: K-Means

Classification Algorithm (Random Forest):

Load Data:
Load the dataset, such as the Iris dataset, containing features (attributes) and the target variable (class labels).

Data Splitting:
Split the dataset into training and testing sets to train the model on one subset and evaluate its performance on another.

Feature Engineering (Optional):
Perform feature engineering if needed, such as handling missing values, scaling features, or encoding categorical variables.

Model Training:
Choose a classification algorithm, like Random Forest, and train the model on the training set.

Model Evaluation:
Use the trained model to make predictions on the testing set and evaluate its performance using metrics like accuracy, precision, recall, or F1-score.

Tuning (Optional):
Fine-tune hyperparameters if necessary to improve the model's performance.

Prediction (Optional):
Use the trained model to make predictions on new, unseen data.

Clustering Algorithm (K-Means):

Load Data:
Load the dataset, ensuring it contains features for clustering.

Feature Engineering (Optional):
Perform feature engineering if needed, such as handling missing values, scaling features, or encoding categorical variables.

Dimensionality Reduction (Optional):
Apply dimensionality reduction techniques, like PCA, to visualize the data if it has more than two dimensions.

Model Training:
Choose a clustering algorithm, like K-Means, and apply it to the dataset.

Number of Clusters:
Specify the number of clusters (K) based on the problem or use techniques like the elbow method to determine an optimal value.

Cluster Assignment:
Assign each data point to a cluster based on the algorithm's output.

Visualization (Optional):
Visualize the clusters, especially if dimensionality reduction was applied, to gain insights into the data distribution.

Interpretation:
Analyze the characteristics of each cluster and interpret the results in the context of the problem.
These steps provide a general outline for applying classification and clustering algorithms to a dataset.

Python Implementation code for the above is:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
from sklearn.decomposition import PCA

# Load the Iris dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
column_names = ["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]
iris_data = pd.read_csv(url, header=None, names=column_names)

# Separate features and target variable
X = iris_data.iloc[:, :-1]
y = iris_data["class"]

# Split the data into training and testing sets for classification
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Classification: Random Forest
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)

# Evaluate the classification model
accuracy = accuracy_score(y_test, y_pred)
print(f"Classification Accuracy: {accuracy:.2f}")

# Clustering: K-Means
# Reduce the dimensionality for visualization purposes
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(X)

# Visualize the clusters
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans_labels, cmap="viridis")
plt.title("K-Means Clustering of Iris Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Q9. Write all programs that were performed in the Lab.

1. Iris Dataset

```
import pandas as pd from sklearn
import datasets from sklearn.model_selection
import train_test_split from sklearn.preprocessing
import LabelEncoder from sklearn.linear_model
import LogisticRegression
import warnings warnings.filterwarnings('ignore')
iris, target = datasets.load_iris(return_X_y=True, as_frame=True)
le = LabelEncoder()
target = le.fit_transform(target)
X_train, X_test, Y_train, Y_test = train_test_split(iris, target, random_state=42)
logreg = LogisticRegression(max_iter = 1000)
logreg.fit(X_train, Y_train) logreg.score(X_test, Y_test)
```

2. California Housing Dataset

```
import seaborn as sns from sklearn
import datasets from sklearn.model_selection
import train_test_split from sklearn.linear_model
import LinearRegression from sklearn.ensemble
import RandomForestRegressor
housing, target = datasets.fetch_california_housing(return_X_y=True, as_frame=True)
X_train, X_test, Y_train, Y_test = train_test_split(housing, target, random_state=42)
rf = RandomForestRegressor()
rf.fit(X_train, Y_train)
rf.score(X_test, Y_test)
```

3. MNIST Dataset

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt from tensorflow.python.keras.callbacks
import ReduceLROnPlateau
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience=2,
verbose=1, factor=0.5, min_lr=0.00001)
callbacks = [learning_rate_reduction]
mnist = tf.keras.datasets.mnist (x_train, y_train), (x_test, y_test) = mnist.load_data() x_train,
x_test = x_train / 255.0, x_test / 255.0
model = tf.keras.models.Sequential([ tf.keras.layers.Flatten(), tf.keras.layers.Dense(256,
activation=tf.nn.relu), tf.keras.layers.Dropout(0.2), tf.keras.layers.Dense(10,
activation=tf.nn.softmax) ])
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(x_train, y_train, epochs=30, validation_data=(x_test, y_test), callbacks=[callbacks], verbose=2)
hist.history['accuracy'][-1]
hist.history['val_accuracy'][-1]
```