* **[Linear Separability](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning" \l "Linear%20Separability)**
  + └── [**Description**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Description)
    - └── [When classes can be separated by a straight line or hyperplane in feature space](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#When%20classes%20can%20be%20separated%20by%20a%20straight%20line%20or%20hyperplane%20in%20feature%20space)
  + ├── [**Techniques**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Techniques)
    - └── [**Perceptron**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Perceptron)
      * └── [Algorithm](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Algorithm)
      * ├── [Applications](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Applications)
    - ├── [**Support Vector Machines (SVM)**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Support%20Vector%20Machines%20(SVM))
      * │ └── [Algorithm](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Algorithm)
      * ├── [Applications](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Applications)
* [**Non-linear Separability**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Non-linear%20Separability)
  + └── [**Description**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Description)
    - └── [When classes cannot be separated by a straight line or hyperplane in feature space](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#When%20classes%20cannot%20be%20separated%20by%20a%20straight%20line%20or%20hyperplane%20in%20feature%20space)
  + ├── [**Techniques**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Techniques)
    - └── [**Kernel Methods**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Kernel%20Methods)
      * └── [**Description**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Description)
        + └── [Techniques that map data into higher-dimensional spaces where it becomes linearly separable](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Techniques%20that%20map%20data%20into%20higher-dimensional%20spaces%20where%20it%20becomes%20linearly%20separable)
      * ├── [**Subcategories**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Subcategories)
        + └── [**Polynomial Kernel**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Polynomial%20Kernel)

    └── [**Description**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Description)

     └── [Maps data into a higher-dimensional space using polynomial functions](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Maps%20data%20into%20a%20higher-dimensional%20space%20using%20polynomial%20functions)

* + - * + ├── [**RBF Kernel**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#RBF%20Kernel)

    └── [**Description**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Description)

     └── [Maps data into a higher-dimensional space using radial basis functions](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Maps%20data%20into%20a%20higher-dimensional%20space%20using%20radial%20basis%20functions)

* + - ├── [**Decision Trees**](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Decision%20Trees)
      * │ └── [Algorithm](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Algorithm)
      * ├── [Applications](https://explorer.globe.engineer/?q=Class+Separability+in+Machine+Learning#Applications)

**Linear Separability**

**Description:** Linear separability refers to the condition where classes can be separated by a straight line or hyperplane in feature space.

**Techniques:**

1. **Perceptron:**
   * **Algorithm:** The Perceptron algorithm is a binary linear classifier that iteratively learns a separating hyperplane to divide the input into two classes.
   * **Applications:** It is used for binary classification tasks where the data is linearly separable.
2. **Support Vector Machines (SVM):**
   * **Algorithm:** SVM is a linear classifier that finds the maximum margin separating hyperplane for binary classification tasks.
   * **Applications:** It is used for binary classification tasks and is particularly effective when the data is linearly separable.

**Non-linear Separability**

**Description:** Non-linear separability occurs when classes cannot be separated by a straight line or hyperplane in feature space.

**Techniques:**

1. **Kernel Methods:**
   * **Description:** Kernel methods are techniques that map data into higher-dimensional spaces where it becomes linearly separable.
   * **Subcategories:**
     + **Polynomial Kernel:**
       - **Description:** It maps data into a higher-dimensional space using polynomial functions.
     + **RBF Kernel:**
       - **Description:** It maps data into a higher-dimensional space using radial basis functions.
2. **Decision Trees:**
   * **Algorithm:** Decision trees are a non-linear classification algorithm that recursively splits the data based on feature thresholds to create a decision structure.
   * **Applications:** They are used for both classification and regression tasks, particularly when the data is not linearly separable.

The use of kernel methods, such as the polynomial and RBF kernels, allows for the transformation of data into higher-dimensional spaces where linear separation becomes possible. Additionally, decision trees provide a non-linear approach to classification, making them suitable for scenarios where linear separability is not achievable.

In order to determine the appropriate technique for ensuring class separability when predicting the controversiality of Reddit comments using topic modeling (LDA), sentiment analysis (VADER), and user karma as inputs in a supervised learning framework (such as SVM or Naive Bayes), you will first need to assess whether your data is linearly separable or non-linearly separable. Here's how you might proceed:

1. **Data Preprocessing**: Start by preprocessing your data. This includes tokenizing the text from Reddit comments, removing stop words, and performing stemming or lemmatization. Then, convert this processed text into numerical features using techniques like Bag of Words, TF-IDF, or word embeddings. Compute sentiment scores using VADER and include user karma as another input feature.
2. **Visualize Data Distribution**: Plot the distribution of your dataset in the feature space to get an initial sense of its separability. You may want to utilize dimensionality reduction techniques like PCA for visualization purposes if you have high-dimensional data. If the data appears to be linearly separable, consider applying linear techniques; otherwise, move on to non-linear approaches.
3. **Apply Linear Techniques (if applicable)**: Try training a simple logistic regression, SVM with a linear kernel, or perceptron model. Evaluate their performance using cross-validation techniques and metrics like accuracy, precision, recall, F1 score, and AUC-ROC. If these models perform reasonably well, then your data is likely linearly separable. However, if they struggle to achieve satisfactory results, try moving on to non-linear techniques.
4. **Apply Non-linear Techniques (if necessary)**: Consider employing kernel methods like Polynomial Kernels or Radial Basis Function (RBF) Kernels within your SVM model. These allow you to transform the data into higher dimensions, potentially enabling better class separability. Alternatively, train decision tree-based algorithms like Random Forests or Gradient Boosted Decision Trees. Compare their performance against linear techniques and select the best one based on evaluation metric values.
5. **Feature Engineering**: Throughout this process, explore various ways of creating new engineered features that could help improve class separability. For instance, combining different existing features, incorporating interaction terms between features, or adding derived statistical measures could all prove beneficial. Reassess class separability after each round of feature engineering until you find an optimal set of features.
6. **Model Tuning & Ensembling**: Fine-tune the parameters of chosen models using grid search or randomized search techniques, focusing on improving generalizability via regularization strategies like L1 or L2 penalties. Once satisfied with individual model performances, combine predictions through stacking or voting ensembles to further enhance overall prediction quality.

To specifically address the aspect of class separability in your project involving using topic modeling (LDA), sentiment analysis (VADER), and user karma as input features for predicting controversiality of Reddit comments using SVM or Naïve Bayes models in a supervised learning setting, follow these outlined steps below:

1. **Inspect Data Distributions**
   * Assess the distribution of your target variable—controversy level—to check for balance among categories. Address imbalance issues using oversampling, undersampling, or SMOTE techniques before continuing.
   * Generate univariate histograms and box plots for input features, examining the spread, center, and shape of the distributions. Verify the presence of any noticeable clusters or gaps between classes.
   * Produce scatterplot matrices or parallel coordinate charts displaying multidimensional views of feature interactions. Look for trends, overlaps, and evident separations.
2. **Compute Separability Metrics**
   * Quantify class separability using distance-based metrics, such as Euclidean distances, Mahalanobis distances, or Bhattacharyya coefficients. Higher values indicate greater separability.
   * Investigate angular separation measures, calculating angles formed between pairs of vectors originating from the mean points of each category. Smaller angles imply lower separability.
   * Utilize overlap indices measuring the amount of shared volume among class density estimates. Lower volumes suggest clearer distinctions between classes.
3. **Linearity Assessment**
   * Determine linear separability by attempting to separate data points belonging to distinct classes using lines, planes, or hyperplanes. Count the number of misclassifications to estimate error rates associated with linear boundaries.
   * Estimate the Fisher criterion, describing the ratio of interclass variance to intraclass variance. Values above unity generally correspond to acceptable linear separability conditions.
4. **Transform Input Space (If Necessary)**
   * Should linear separability fail, attempt to apply dimensionality reduction techniques like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or Multidimensional Scaling (MDS) to discover alternative low-dimensional representations capturing inherent structures facilitating improved linear separability.
   * Employ nonlinear transformations utilizing kernel tricks, raising original data into higher-dimensional spaces. Common examples include polynomial and Gaussian kernels.
5. **Model Selection Based On Separability Results**
   * Given the outcomes from the preceding analyses, choose the most promising machine learning algorithms capable of exploiting discovered characteristics promoting increased class separability. Prioritize methods exhibiting strong theoretical foundations addressing underlying separability assumptions identified earlier.
6. **Evaluation and Validation Of Chosen Algorithms**
   * Following selection, split data into training, validation, and testing sets to prevent overfitting and underestimation biases. Retrain chosen algorithms on increasingly large fractions of data (starting with small proportions), validating intermediate outputs against reserved test subsets.
   * Track changes in assessment statistics throughout successive iterations, pinpointing stages demonstrating peak performance. Document variations attributed to increasing amounts of utilized data alongside effects resulting from applied optimization procedures.
7. **Optimization And Hyperparameter Adjustment**
   * While tracking progress made towards enhancing class separability, experiment with diverse parameter settings affecting the behavior of chosen algorithms. Focus efforts on exploring tradeoffs between bias and variance, balancing complexity and interpretability concerns.
   * Optimize regularization techniques, controlling capacity expansion, preventing ill-posed problems, and mitigating overfitting risks. Regularizers include L1 (Lasso), L2 (Ridge), Elastic Net, and Dropout constraints.
8. **Ensemble Strategies (Optional)**
   * When confronted with persistently challenging class separability challenges despite exhaustive attempts, entertain possibilities entailing fusion of complementary output signals sourced from disparate algorithmic pipelines.
   * Design customized meta-learning schemes leveraging unique strengths possessed by previously considered candidate algorithms. Formulate novel objective functions incentivizing accurate consensus judgments amidst competing constituent estimators.
9. **Reporting Findings**
   * Summarize empirical evidence supporting claims concerning enhanced class separability achieved through executed processes. Provide concise yet informative explanations illustrating critical insights gained during development phases. Clearly articulate benefits accrued from proposed solutions relative to baseline alternatives.