# Q1

## Data Sampling and Preprocessing

* The dataset, "Train\_rev1.csv," comprising various job-related attributes, was initially loaded and a subset of 2,500 samples was selected randomly to manage computation efficiency.
* The **SalaryNormalized** field was used to determine salary ranges, with the 75th percentile as the threshold to label salaries as high or low.

## Base BV Model

* A baseline model was established using **TfidfVectorizer** for text feature extraction from the **FullDescription** field, limiting the features to 5,000 and excluding stopwords.
* The **MultinomialNB** classifier was chosen for its suitability for text classification tasks.
* The baseline model achieved an accuracy of 0.758 with a confusion matrix indicating the model's performance on both classes.

Accuracy: 0.758

Confusion Matrix:

[[365 6]

[115 14]]

Model Refinement

Random Search for Hyperparameter Optimization

* A **RandomizedSearchCV** was employed to tune both the vectorizer and classifier parameters, exploring a range of values for max\_df, min\_df, ngram\_range, and alpha.
* The refined model demonstrated improved accuracy (0.77), validating the effectiveness of hyperparameter tuning.

Best parameters:

{'clf\_\_alpha': 0.08761761457749352, 'tfidf\_\_max\_df': 0.8005575058716043, 'tfidf\_\_min\_df': 3, 'tfidf\_\_ngram\_range': (1, 2)}

New Accuracy: 0.77

New Confusion Matrix:

[[322 49]

[ 66 63]]

## Ensemble model with Random Forest

* An exploration into using **RandomForestClassifier** with tuned parameters via **RandomizedSearchCV** was conducted to compare against the Naïve Bayes model.
* The Random Forest model showed competitive accuracy (0.764) but did not significantly outperform the Naïve Bayes model.

RandomForest Tuned Accuracy: 0.764

Confusion Matrix for the Tuned RandomForest Model:

[[367 4]

[114 15]]

Ensemble Accuracy: 0.764

## Apply Feature Engineering and test the best model we get above

* An enhancement to the preprocessing step was introduced by combining multiple text attributes (**FullDescription**, **Title**, **Company**) into a single feature to enrich the model's input.
* The feature-engineered Naïve Bayes model yielded an accuracy of 0.782, demonstrating the impact of comprehensive text features on model accuracy.

Text-based NB Model Accuracy: 0.782

Text-based NB Model Confusion Matrix:

[[325 46]

[ 63 66]]

The best-performing model is the feature-engineered Naïve Bayes model which yielded an accuracy of 0.782, demonstrating the impact of comprehensive text features on model accuracy.

**Parameters:**

memory: None

steps: [('text\_preprocessor', FunctionTransformer(func=<function <lambda> at 0x00000296A085B2E0>)), ('tfidf', TfidfVectorizer(max\_df=0.8005575058716043, min\_df=3, ngram\_range=(1, 2),

stop\_words=['i', 'me', 'my', 'myself', 'we', 'our', 'ours',

'ourselves', 'you', "you're", "you've", "you'll",

"you'd", 'your', 'yours', 'yourself', 'yourselves',

'he', 'him', 'his', 'himself', 'she', "she's",

'her', 'hers', 'herself', 'it', "it's", 'its',

'itself', ...])), ('classifier', MultinomialNB(alpha=0.08761761457749352))]

verbose: False

text\_preprocessor: FunctionTransformer(func=<function <lambda> at 0x00000296A085B2E0>)

tfidf: TfidfVectorizer(max\_df=0.8005575058716043, min\_df=3, ngram\_range=(1, 2),

stop\_words=['i', 'me', 'my', 'myself', 'we', 'our', 'ours',

'ourselves', 'you', "you're", "you've", "you'll",

"you'd", 'your', 'yours', 'yourself', 'yourselves',

'he', 'him', 'his', 'himself', 'she', "she's",

'her', 'hers', 'herself', 'it', "it's", 'its',

'itself', ...])

classifier: MultinomialNB(alpha=0.08761761457749352)

text\_preprocessor\_\_accept\_sparse: False

text\_preprocessor\_\_check\_inverse: True

text\_preprocessor\_\_feature\_names\_out: None

text\_preprocessor\_\_func: <function <lambda> at 0x00000296A085B2E0>

text\_preprocessor\_\_inv\_kw\_args: None

text\_preprocessor\_\_inverse\_func: None

text\_preprocessor\_\_kw\_args: None

text\_preprocessor\_\_validate: False

tfidf\_\_analyzer: word

tfidf\_\_binary: False

tfidf\_\_decode\_error: strict

tfidf\_\_dtype: <class 'numpy.float64'>

tfidf\_\_encoding: utf-8

tfidf\_\_input: content

tfidf\_\_lowercase: True

tfidf\_\_max\_df: 0.8005575058716043

tfidf\_\_max\_features: None

tfidf\_\_min\_df: 3

tfidf\_\_ngram\_range: (1, 2)

tfidf\_\_norm: l2

tfidf\_\_preprocessor: None

tfidf\_\_smooth\_idf: True

tfidf\_\_stop\_words: ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

tfidf\_\_strip\_accents: None

tfidf\_\_sublinear\_tf: False

tfidf\_\_token\_pattern: (?u)\b\w\w+\b

tfidf\_\_tokenizer: None

tfidf\_\_use\_idf: True

tfidf\_\_vocabulary: None

classifier\_\_alpha: 0.08761761457749352

classifier\_\_class\_prior: None

classifier\_\_fit\_prior: True

classifier\_\_force\_alpha: warn

## Top 10 indicative words for Low Salary:

sales

experience

manager

work

role

team

business

skills

working

care

## Top 10 indicative words for High Salary:

business

experience

project

manager

development

management

team

senior

role

client

# Q2

**Lemmatization/Stemming:** Apply lemmatization or stemming to reduce words to their base or root form, potentially improving the model's ability to generalize from the training data.

**Model Stacking:** Combine predictions from multiple models (not just NB) through stacking. Train a meta-classifier on the predictions made by base classifiers on a hold-out set to capture different aspects of the data.

**Bayesian Optimization:** Use Bayesian optimization techniques that can be more efficient than grid or random search by focusing on areas of the parameter space most likely to improve model performance.

**Custom Stop Words:** Beyond standard stop words, identify and remove domain-specific stop words that are not informative for salary prediction.