ML Project Report

# Problem Setup

The goal of this project is to predict the sale prices of bulldozers based on various features such as model information, product class, and sale date.

The dataset used for this project is obtained from a CSV file named "Train.csv." Whilst there were other files in the dataset we downloaded from Kaggle as well, we used Train.csv for both training and validation (with splits, of course), as the “Valid.csv” and “Test.csv” files did not have the target “SalePrice” columns that we required for evaluation.

To keep things consistent and to allow a comparison and analysis of the different approaches used, we used the same evaluation metric across all notebooks: the RMSLE. The RMSLE does not severely penalize large errors, which helps take into account anomalies, thus making it a good evaluation metric for price predictions where such outliers are common. This was also the metric used in the original Kaggle competition we got the dataset from.

**Note:** To keep this report succinct and readable, it does not contain in-depth justifications for all actions taken (such as particular data cleaning steps or hyperparameter selection). For more details on any of this, please check out the markdown comments in the individual Jupyter notebooks. :)

Neural Networks

**Data Preprocessing:** Checked and handled missing values, dropping columns with a significant number of NaN values. Created new features from the saledate column, breaking it down into day, month, and year. Utilized label encoding for categorical variables, converting them into a format suitable for machine learning models. Dropped unnecessary columns to streamline the dataset.

**Data Splitting and Scaling:** Split the dataset into training and testing sets using the train\_test\_split function. Scaled the features using MinMaxScaler to ensure uniformity and enhance model performance.

**Model Architecture:** Implemented a neural network model (ComplexModel) using PyTorch, consisting of multiple fully connected layers with ReLU activation functions. Defined a custom loss function (RMSLELoss) to address the specific requirements of the regression task.

**Training:** Used the Adam optimizer with a learning rate of 0.01 to train the model. Iterated through 10,000 epochs, continually updating the model parameters to minimize the RMSLE loss.

**Findings:** The model demonstrated a consistently decreasing training loss over the epochs, indicating effective learning. Evaluated the model on the test set, achieving a test loss of 0.34, suggesting good generalization performance.

**Evaluation:**

We chose the RMSLE loss as the evaluation metric, as it is well-suited for regression tasks, penalizing underestimation and overestimation of sale prices equally. The achieved test loss of 0.34 indicates that the model successfully learned the underlying patterns in the data and generalizes well to unseen samples.

KNN

**Data Cleaning:** We began by cleaning our dataset for machine price prediction. This involved dropping columns that had more than 30% missing values. Additionally, we took care of missing data by filling in the null values with the median value.

**Insights on Using KNN:** K-Nearest Neighbors is a useful algorithm for predicting prices, especially when dealing with non-linear relationships in the data. However, during our implementation, we noticed and confirmed that KNN is sensitive to the scale of features. To address this, we took the necessary step of standardizing or normalizing features. This ensures that features with larger scales don't disproportionately influence the distance calculations.

**Challenges with the Curse of Dimensionality:** One crucial aspect to consider is the "Curse of Dimensionality." In simpler terms, as we deal with more features or dimensions, the concept of proximity between data points becomes less meaningful. This can have a negative impact on the performance of KNN. Essentially, in high-dimensional spaces, the distance between points increases exponentially as the number of features grows. This is an important consideration as it can affect the accuracy of our predictions, especially as our dataset becomes more complex. That’s why I considered dropping numerous columns in the notebook.

**Overall Considerations:** While KNN has its strengths, such as capturing non-linear patterns, it's essential to be mindful of its sensitivity to feature scale and the potential challenges introduced by high-dimensional datasets. Despite these considerations, we implemented protocols to standardize features and carefully handle missing data, aiming to optimize the performance of KNN for our specific machine price prediction task.

# Linear Regression & Random Forest

**Data Preprocessing:** Initial steps included loading data and parsing the 'saledate' column into features like 'saleYear,' 'saleMonth,' and 'saleDay,' as well as 'saleDayOfWeek' and 'saleDayOfYear' for detailed temporal analysis. We efficiently tackled missing data by imputing numeric values and formatting categorical variables, ensuring data integrity and robustness for modeling

**Exploratory Data Analysis (EDA):** An in-depth EDA revealed data patterns, particularly within the “saledate” attribute. This was split into year, month, and day for detailed time analysis. String data was also transformed into ordered categorical variables, improving analysis

**Model Development and Rationale:** The project began with a Linear Regression model for its ease and clarity. Standardizing numeric features improved its performance. Later, a Random Forest Regressor was chosen for its non-linear capabilities and overfitting resistance, leading to better predictive accuracy

**Model Development and Evaluation:** Linear Regression was the main model used, with standard scaling for numeric data. It was evaluated using RMSE, yielding scores of about 8000 for training and 8500 for validation. RMSLE was around 0.5. A Random Forest Regressor was also tested, showing an MAE of 2923 and RMSLE of 0.1432. Its high R^2 value of 0.9598 indicates it explains nearly 95.98% of the training set variance, outperforming the linear model

**Conclusion and Insights:** These successfully created predictive models for bulldozer pricing and identified key price factors. It aids stakeholders in making informed pricing decisions. The study compared Linear Regression and Random Forest models, noting their pros and cons. Future work includes adding features, trying advanced models, and fine-tuning the Random Forest for better price predictions.

Decision Trees

**Data Processing and Feature Selection:**

**Null Values :** In the dataset, some features were mostly missing values. I've discarded the features on a set threshold (>40% missing data).

**Correlation :** I have selected the numerical features by analyzing the correlation matrix. The features that were very correlated have been discarded to avoid redundancy.

**Non-Numerical :** While differentiating the numerical/non-numerical features, I have analyzed the uniqueness of each feature. For instance, if a feature has 3 numerical unique values, I have treated those as classification features.

**Scaling :** Upon reading the content, I found that tree based methods are not affected by scaling. That's why I have not used feature scaling.

**Missing Values :** Since some features had a high number of missing values, I have not filled them randomly or with a fixed value. Instead, I have experimented with KNN and Regression methods to fill values.

**New Features :** The date feature proved to be of high value. That's why I have splitted the date into months, quarters, days for new features.

**Memory Saving :** Our dataset contained more than 4 lac entries with 50+ features. After feature selection, I have quantized the dataset for faster operations.

**Model**: I have used the Decision Tree regressor for the problem. I have tuned the hyper-parameters by Grid Search to start with. Later, I have used techniques like pruning and regularization to avoid overfitting. Also, I limited the parameters like depth and max-features (features used to make a decision for splits). This helps the model avoid relying too much on one feature and predict using more feature

**Evaluation**: For the evaluation of the model, I have used the metric used in the official dataset documentation **RMSLE**. For better understanding of the model, I used visualizations of the tree. These included generic decision paths for making predictions and the importance of each feature in making the decision. The pruned model gives an RSMLE of 0.4 and the model focuses more on the features of the actual product for predicting the sale price.

# LGBM (Light Gradient Boosting Machine)

**Data Preprocessing:** The ‘saledate’ column was split into saleYear and saleMonth. Columns that had 'None or Unspecified' entries were identified, and all missing values in these columns were replaced by 'None or Unspecified' (as the existence of your entries implied that it is okay for there to be unspecified information in the column). Columns with >75% missing values were dropped. Specific columns that seemed useless were dropped, in particular: 'datasource', 'auctioneerID', 'ProductGroupDesc'. Numerical columns were imputed with median values. Categorical columns were imputed with most frequent values. Categorical columns were label encoding.

**Model:**

LightGBM, a gradient boosting framework (similar to XGBoost), is used here for building our model. LGBM is known for its efficiency in handling large datasets and speed of execution, and so it was expected to perform well here. The model was trained and evaluated on “Train.csv”, with an 80-20 split.

Note: We also tried XGBoost previously, but it was not performing as well, as the model kept overfitting drastically. We initially thought this may have been due to the curse of dimensionalities brought on by one-hot encoding categorical columns, but even when we removed many categorical columns (especially those with high cardinality), and even experimented with label encoding, it still overfit. Thus, we experimented with LGBM.

**Evaluation:**

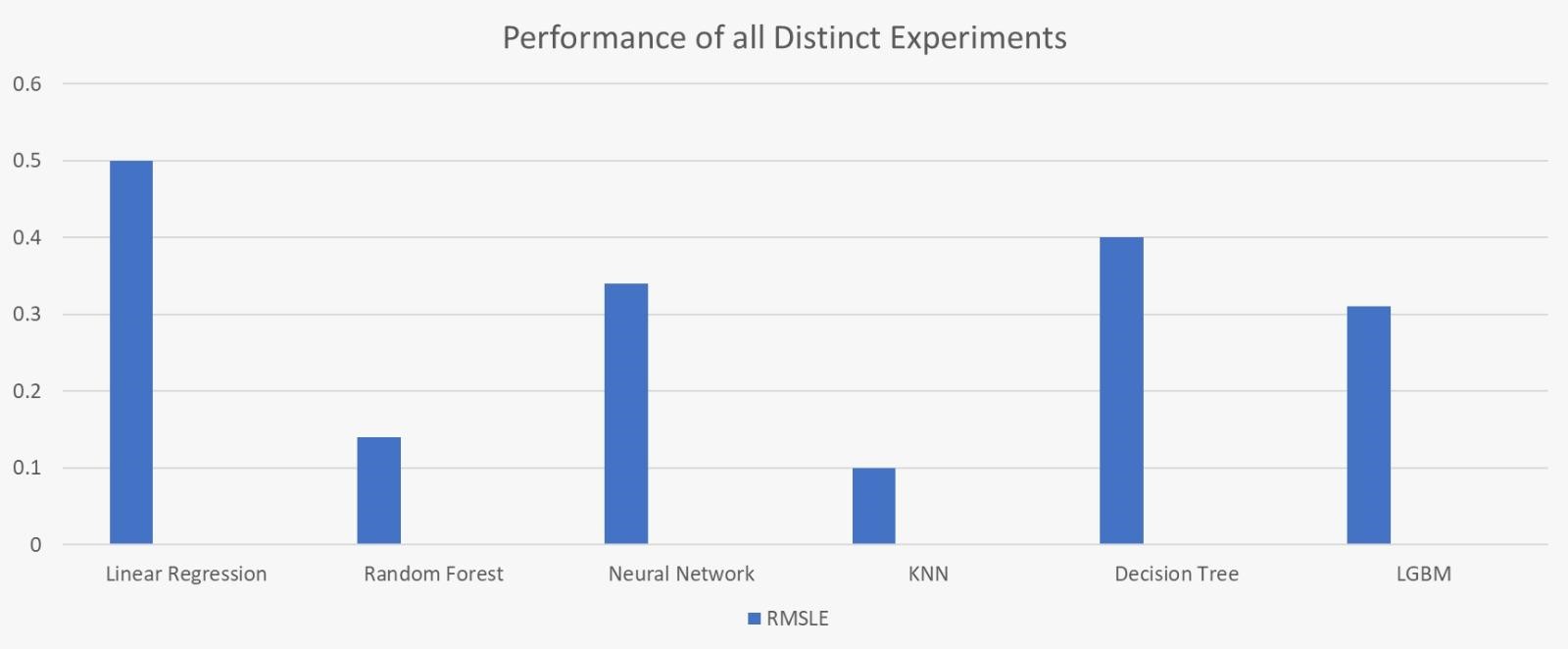
The model fit quite well, even with initial attempts with minimal data cleaning. It generalized well and did not seem to overfit at any point, and selected features very effectively. We proceeded to experiment with more data cleaning (which is mentioned above), removing many columns and combinations, expecting it to improve performance. However, barely any columns made it improve performance, with most removals resulting in slight dips of performance. We felt the model was able to extract some meaningful information even from features we had deemed useless, and thus we let them stay in (with the exception of 3 that we did remove).

Eventually, we ended up with an **RMSLE** of **0.3159.**

**Conclusion**

**Insights:**

Our diverse models, from Neural Networks to Decision Trees and LGBM, provided valuable insights into bulldozer price prediction. Despite strong performance, limitations in dataset size and metric choice exist. We recommend exploring additional features, fine-tuning hyperparameters, and continuous model updates for enhanced accuracy.



**Limitations**:

Dealing with the dataset's sheer size and complexity poses challenges in capturing every factor influencing bulldozer prices. While RMSLE suits our needs, it may not fully capture the intricacies of our model's performance. It's important to acknowledge these limitations for a more nuanced understanding and continual improvement in our predictive approach.

**Recommendations**:

We suggest exploring additional features to boost predictive accuracy. Fine-tune hyperparameters for optimal model performance and consider experimenting with ensemble models. Regularly update models with new data for adaptability to evolving market conditions, ensuring sustained relevance and improved performance over time.