INDUSTRIAL COPPER MODELING

"Industrial Copper Modeling" is a machine learning project aimed at addressing challenges in the copper industry, particularly related to sales and pricing data. Here's an explanation of the best approach for this task, along with detailed steps and methodologies that align with the supervised predictive model development guidelines.

DOMAIN: Manufacturing

Tech Stacks Employed:

- Python scripting,
- Pandas,
- Data Preprocessing,
- Visualization,
- EDA,
- Scikit learn: Multi-Linear Regression, Decision Tree Classifier, Logistic Regression Classifier, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier
- Streamlit.

Approach for the Industrial Copper Modeling Task:

I. Project Understanding and Problem Statement

Objective: Develop machine learning models to predict the selling price and classify the lead status (Won/Lost) in the copper industry.

Challenges:

- Dealing with skewed and noisy data.

- Need for data normalization and handling outliers.

- Creating a user-friendly interface to predict outcomes.

II. Data Understanding and Preprocessing

Data Characteristics:

- The dataset includes various columns like `id`, `item_date`, `quantity tons`, `customer`, `country`, `status`, `item type`, `application`, `thickness`, `width`, `material_ref`, `product_ref`, `delivery date`, and `selling_price`.

Data Preprocessing Steps:

1. Exploring Data Skewness and Outliers:

- Visualize: Use box plots, histograms, and scatter plots to identify outliers and skewed data.

- Treat Outliers: Apply methods like the Interquartile Range (IQR) or Isolation Forest to handle outliers.

- Handle Skewness: Transform skewed variables using techniques such as log transformation, Box-Cox transformation, or square root transformation.

2. Data Cleaning:

- Handle Missing Values: Use mean, median, or mode imputation for missing values. Alternatively, use more sophisticated methods like KNN imputation if applicable.

- Remove or Replace Invalid Data: Convert invalid `Material_Reference` values (e.g., starting with '00000') to null or handle appropriately.

3. Feature Engineering:

- Encode Categorical Variables: Use one-hot encoding for nominal data or label encoding for ordinal data.

- Create New Features: Aggregate existing features to create more informative variables. For example, combine `thickness` and `width` to create a new feature representing the area.

4. Data Normalization:

- Scale Continuous Variables: Apply standard scaling (StandardScaler) or min-max scaling (MinMaxScaler) to ensure that the data is on a comparable scale.

III. Exploratory Data Analysis (EDA)

EDA Objectives:

- Understand data distributions and relationships.

- Identify patterns and correlations between features.

EDA Techniques:

- Visualize Distributions: Use Seaborn's `distplot` and `boxplot` to check the distribution of continuous variables.

- Correlation Analysis: Use a heatmap to identify highly correlated features and decide which to drop.

- Explore Relationships: Use scatter plots to explore relationships between features and the target variable (`selling_price`).

IV. Model Building and Evaluation

Modeling Steps:

1. Split the Data:

- Divide the dataset into training and testing sets using an 80-20 split or similar ratio.

2. Regression Model for Selling Price:

- Select Models: Consider tree-based models like Random Forest, Gradient Boosting, or XGBoost, which handle non-linear relationships and skewed data well.

- Train and Evaluate: Train the models using the training data and evaluate them using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 on the test set.

- Hyperparameter Tuning: Use GridSearchCV or RandomizedSearchCV for optimizing model parameters.

3. Classification Model for Lead Status:

- Select Models: Consider models like Logistic Regression, Random Forest Classifier, or XGBoost Classifier.

- Train and Evaluate: Evaluate using metrics like accuracy, precision, recall, F1 score, and the AUC-ROC curve.

- Hyperparameter Tuning: Optimize the model using techniques like cross-validation.

V. Creating a Streamlit Application

Streamlit Implementation:

1. Set Up Input Fields:

- Create fields for each input feature, allowing users to enter new data for prediction.

2. Load Models and Transformations:

- Use pre-saved models and transformations (e.g., scalers, encoders) to preprocess input data.

- Load models using `pickle` or similar libraries.

3. Predict and Display Results:

- Based on user input, preprocess the data, make predictions, and display the results.
- For regression, show the predicted `selling price`.
- For classification, show the `status` as either "Won" or "Lost".

4. User Interface:

- Design a simple and intuitive interface that allows users to easily input data and view predictions.

VI. Best Practices and Tips

Code Quality and Maintenance:

- Modular Code: Write code in functions or classes to improve readability and reusability.
- Compliance: Follow Python coding standards as per PEP 8.

Evaluation Metrics:

- Accuracy and Performance: Use the appropriate metrics to evaluate model performance.
- Scalability: Ensure that the models and application can handle increased data volumes and user requests.

By following these steps and best practices, you can effectively address the problem statement provided in the PDF document and create a robust solution for predicting selling prices and classifying leads in the copper industry.