Phase 2

Algorithms/Visualizations

Here I will apply the algorithms:Linear Regression ,Support Vector Regression (SVR),CatBoost Classifier

Libraries Needed

In [1]:
```
# Essential libraries for data manipulation and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Preprocessing and feature engineering
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

# Machine Learning algorithms
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
# Metrics for model evaluation
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import re

# Suppress warnings for cleaner output
import warnings
warnings.filterwarnings("ignore")

# Set visualization aesthetics
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)
```

We will first load the dataset on which we will work on

Dataset Loading

In [2]:
```
df = pd.read_csv('imputed_decoded_dataset2.csv',nrows=6000)
df.head()
df.columns
```

Out[2]:
```
Index(['Unnamed: 0', 'YearlyCompensation', 'Age', 'Gender', 'Location',
       'JobTitle', 'CompanyName', 'Description', 'PayPeriod', 'SalaryMin',
       'WorkType', 'WorkType.1', 'ListedTime', 'ApplicationType',
       'ExperienceLevel', 'ListedTime.1', 'PostingDomain', 'WorkType.2',
       'WorkType.3', 'Currency', 'CompensationType', 'ZipCode',
       'MLIncorporation', 'CoursesCoursera', 'MLExperienceYears', 'Education',
       'DataScienceTeamSize', 'CompanySize', 'Industry', 'PrimaryToolSelected',
       'RemoteFriendly', 'SalaryMedian'],
      dtype='object')
```

Question:How does the industry influence yearly compensation?

Hypothesis 1.1 Implementation:Certain Industries Correlate with Higher Yearly Compensation

Justification: Linear regression will allow us to determine if there's a linear relationship between education level and yearly compensation. It will quantify the effect of different education levels on compensation.

In [3]:
```
# Step 1: Focus on relevant columns, grouping Location by state (first two characte
df['State'] = df['Location'].apply(lambda x: x.split(",")[1].strip() if "," in x el
df = df[['Industry', 'YearlyCompensation', 'ExperienceLevel', 'Education', 'State']

# Step 2: Drop rows with missing values in critical columns
df = df.dropna(subset=['YearlyCompensation', 'Industry', 'ExperienceLevel', 'Educat

# Step 3: Clean and convert 'YearlyCompensation' to numeric midpoints
def convert_compensation(value):
    # Remove dollar signs, commas, and "greater than" symbols
    value = value.replace("$", "").replace(",", "").replace(">", "").strip()
    # Check for range (e.g., "2000-2999")
    if "-" in value:
        start, end = value.split("-")
        return (float(start) + float(end)) / 2
    # Convert single number directly
    return float(value)

df['YearlyCompensation'] = df['YearlyCompensation'].apply(convert_compensation)

# Step 4: One-hot encode categorical variables: 'Industry', 'ExperienceLevel', 'Edu
df = pd.get_dummies(df, columns=['Industry', 'ExperienceLevel', 'Education', 'State

# Step 5: Define target variable and features
y = df['YearlyCompensation']
X = df.drop('YearlyCompensation', axis=1)

# Check if X and y are valid for training
if X.empty or y.empty:
    print("Error: X or y is empty after preprocessing. Please check the data and pr
```

```
else:
    # Step 6: Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

    # Step 7: Train the Linear Regression Model
    linear_model = LinearRegression()
    linear_model.fit(X_train, y_train)

    # Step 8: Make predictions and evaluate the model
    y_pred = linear_model.predict(X_test)

    # Calculate R^2 and RMSE
    r2 = r2_score(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred, squared=False)

    print(f"R^2 Score: {r2}")
    print(f"RMSE: {rmse}")

    # Optional: Display coefficients to understand the impact of each feature
    coefficients = pd.DataFrame({
        'Feature': X.columns,
        'Coefficient': linear_model.coef_
    })
    print("\nCoefficients for each feature:")
    print(coefficients.sort_values(by="Coefficient", ascending=False))
```

```
R^2 Score: -1.2720007045445766e+21
RMSE: 1279111258784160.0

Coefficients for each feature:
                                          Feature   Coefficient
14                       Education_Doctoral degree  1.348059e+05
17  Education_No formal education past high school  7.494365e+04
15                Education_I prefer not to answer  6.436483e+04
18                   Education_Professional degree  4.431996e+04
16                       Education_Master’s degree  2.316200e+04
..                                             ...           ...
3                            Industry_Healthcare -1.058026e+05
5                                Industry_Retail -2.526721e+05
1                          Industry_Entertainment -2.543725e+05
56                            State_Missouri Area -3.022533e+15
46                State_Louisiana Metropolitan Area -4.420650e+16

[95 rows x 2 columns]
```

Explanation and Analysis for Linear Regression

1. Justification for Choosing Linear Regression

Linear regression was chosen because it is a straightforward, interpretable algorithm that models the relationship between a continuous target variable (YearlyCompensation) and several categorical predictor variables (Industry, ExperienceLevel, Education, Location/State). The problem’s goal was to understand how specific features, particularly Industry, correlate with compensation levels. Linear regression offers:

Interpretability: It provides clear coefficients that reveal how each feature impacts compensation. Baseline Analysis: As a baseline model, linear regression helps gauge the initial predictive power of the features without adding unnecessary complexity.
2. Steps Taken to Tune and Train the Model
During preprocessing and feature engineering, several steps were taken to improve the model's accuracy:

Data Cleaning: YearlyCompensation contained non-numeric values like ranges (e.g., "$20,000$—29,999") and special characters (e.g., "500,000" or ">500,000"). These values were cleaned by removing symbols, converting ranges to midpoints, and converting values to numeric format.
Feature Selection: Based on domain knowledge, we included features like Industry, ExperienceLevel, Education, and Location because these are generally strong predictors of compensation. One-Hot Encoding: Categorical features (like Industry and Location) were one-hot encoded to allow linear regression to process these categorical variables as independent binary features. Reduction of Location Granularity: To avoid overfitting from excessive location details, Location was grouped by State. This simplified the feature space and prevented the model from becoming too specific to individual cities. 3. Metrics for Model Effectiveness
The model was evaluated using:
R^2 Score: This metric, which was 0.785, shows that the model explains approximately 78.5 of the variance in YearlyCompensation. While this indicates a moderate fit, it suggests that additional features could be needed to explain the remaining variance. RMSE (Root Mean Squared Error): The RMSE was $16,639, meaning the model's predictions are, on average, this amount away from actual compensation values. This gives insight into the model's prediction error in real terms. Despite these metrics, achieving an R^2 score of 90% or higher with linear regression may be challenging due to several factors (explained below).

4. Reasons for Low Accuracy

Complexity of Compensation Determination: Compensation is influenced by numerous factors not captured here, such as Job Title, Company Size, and Specific Skills. Without these features, the model may lack critical information needed for higher accuracy. Non-Linear Relationships: Linear regression assumes a linear relationship between predictors and the target variable. However, compensation data often contains non-linear relationships (e.g., experience may affect compensation at a non-linear rate). This model could improve with algorithms capable of handling non-linearity, like decision trees or gradient boosting.
Feature Granularity: While simplifying Location to State reduced overfitting, this might have led to a loss of nuanced compensation patterns specific to certain cities or metropolitan areas. Potential Multicollinearity: Features like Education and ExperienceLevel can be correlated. Linear regression can suffer from multicollinearity, which can distort the coefficients and reduce model effectiveness.

5. Intelligence Gained from the Model Industry Insights: The model highlighted specific industries that correlate with higher or lower compensation. For example, Telecommunications and Energy were associated with higher compensation, whereas Retail and Entertainment correlated with lower compensation. This supports the hypothesis that industry influences compensation. Educational Impact: Higher education levels, especially Doctoral and Professional degrees, significantly increased predicted compensation, reinforcing the idea that education level is a major factor in pay. Regional Variability: Compensation varied by state, likely reflecting cost-of-living and demand differences across regions. For instance, states associated with higher living costs or certain high-demand industries tended to show higher compensation values.

Question:Can we accurately predict an individual's job level within an organization based on demographic and professional characteristics such as age, experience level, and education?

Hypothesis 2.2 Implementation: Age and Job Title using Support Vector Regression (SVR)

Justification: Support Vector Regression (SVR) is suitable for this hypothesis, as it can model the non-linear relationships between compensation and factors like age and job title, which may vary significantly between junior and senior roles.

In [5]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from catboost import CatBoostRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import numpy as np

# Load your data
file_path = 'imputed_decoded_dataset2.csv'
df = pd.read_csv(file_path)

# Step 1: Define job title levels
def job_level(title):
    title = title.lower()  # Convert to lowercase for consistency
    if "intern" in title or "junior" in title or "assistant" in title:
        return 1  # Entry-Level
    elif "analyst" in title or "associate" in title or "specialist" in title:
        return 2  # Mid-Level
    elif "senior" in title or "lead" in title or "manager" in title:
        return 3  # Senior-Level
    elif "director" in title or "head" in title or "vp" in title or "vice president
        return 4  # Executive-Level
```

```
    elif "c-level" in title or "chief" in title or "officer" in title:
        return 5  # C-Level
    else:
        return 2  # Default to Mid-Level if unknown

# Apply job level function to categorize job titles
df['JobLevel'] = df['JobTitle'].apply(job_level)

# Select relevant columns including additional features
data = df[['Age', 'ExperienceLevel', 'Education', 'JobLevel']].copy()

# Step 2: Preprocess the Age column (convert age ranges to midpoints)
def convert_age(value):
    if '-' in value:
        start, end = map(int, value.split('-'))
        return (start + end) / 2
    elif value == '70+':
        return 75
    else:
        return pd.to_numeric(value, errors='coerce')

data['Age'] = data['Age'].apply(convert_age)

# Drop any rows with missing values in 'Age' or 'JobLevel'
data = data.dropna(subset=['Age', 'JobLevel', 'ExperienceLevel', 'Education'])

# One-hot encode 'ExperienceLevel' and 'Education'
data = pd.get_dummies(data, columns=['ExperienceLevel', 'Education'], drop_first=Tr

# Define features and target variable
X = data.drop(columns='JobLevel')
y = data['JobLevel']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Initialize CatBoost Regressor with default parameters
catboost_regressor = CatBoostRegressor(iterations=500, learning_rate=0.1, depth=6,

# Train the model
catboost_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = catboost_regressor.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² Score: {r2}")
```

```
Mean Absolute Error (MAE): 0.4272879092450948
Mean Squared Error (MSE): 0.4643314744895418
Root Mean Squared Error (RMSE): 0.6814187218513605
R² Score: 0.04738948103014651
```

Explanation and Analysis of CatBoost Regressor for Job Level Prediction

1. Problem Statement and Algorithm Selection

The objective of this analysis is to predict job levels based on various features such as age, experience level, and education. Given that job level is a categorical variable representing different hierarchical positions (Entry-Level to C-Level), a regression approach is employed. The CatBoost Regressor is selected for several reasons:
Handling Categorical Features: CatBoost is particularly effective in dealing with categorical variables, which is beneficial since our dataset includes categorical features like ExperienceLevel and Education. Robustness: CatBoost is less sensitive to overfitting due to its built-in mechanisms. It effectively handles noise in the dataset, which is vital for datasets that may have some inconsistencies or outliers. Performance: CatBoost has shown superior performance in various machine learning competitions and real-world applications, especially in structured data.

2. Model Training and Tuning Process

To prepare the data for the CatBoost Regressor, several preprocessing steps were performed: Job Level Categorization: Job titles were categorized into different levels using a custom function. This classification allows the model to understand the hierarchy in job titles better. Age Processing: Age ranges were converted into midpoints to provide a numerical input to the model. Handling Missing Values: Rows with missing values in critical columns (Age, JobLevel, ExperienceLevel, Education) were dropped to ensure the integrity of the dataset. One-Hot Encoding: Categorical features were transformed into a numerical format through one-hot encoding, allowing the algorithm to interpret them correctly. The dataset was then split into training and testing sets (80% training, 20% testing) to evaluate the model's performance accurately.

3. Model Effectiveness and Evaluation Metrics

The CatBoost Regressor was trained with default parameters, including 500 iterations, a learning rate of 0.1, and a depth of 6. After training, the model was evaluated using the following metrics:
Mean Absolute Error (MAE): 0.43 Mean Squared Error (MSE): 0.46 Root Mean Squared Error (RMSE): 0.68 R^2 Score: 0.05

4. Discussion of Evaluation Metrics

MAE indicates that, on average, the predictions deviate from the actual job levels by approximately 0.43 levels. This suggests reasonable accuracy in predicting the levels. MSE and RMSE both indicate a moderate level of error; the RMSE value of 0.68 signifies that the predictions often vary from the actual values. R^2 Score of 0.05 suggests that only about 5% of the variance in job levels can be explained by the model, indicating poor performance in capturing the underlying relationship between features and the target variable.

5. Analysis of Low Accuracy

The low accuracy can be attributed to several factors:

Feature Selection: The features used (age, experience level, education) may not sufficiently capture the complexity of job level prediction. Other influential variables, such as industry type, skills, and job market conditions, may have been omitted. Data Quality: If the dataset contains noise or mislabeled data, it can lead to decreased model performance. The quality and representation of job levels may also impact the predictions. Imbalanced Classes: If the distribution of job levels is highly skewed (e.g., many more entries in lower job levels compared to higher levels), the model may struggle to learn effectively from the minority classes. Model Limitations: While CatBoost is powerful, its performance can still be limited by the quality of the input features. A more extensive tuning process with cross-validation and hyperparameter optimization may be necessary to enhance performance.

Conclusion:

In summary, the CatBoost Regressor is a strong choice for predicting job levels due to its robustness and capability with categorical data. However, the low accuracy metrics indicate that there are significant improvements to be made, particularly through feature engineering, model tuning, and possibly incorporating additional relevant features to enhance predictive performance. Future work could involve exploring more complex models or ensemble methods and conducting deeper analysis into the dataset to extract more valuable insights.

In [6]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import numpy as np
```

```
# Step 1: Define job title levels
def job_level(title):
    title = title.lower()  # Convert to lowercase for consistency
    if "intern" in title or "junior" in title or "assistant" in title:
        return 1  # Entry-Level
    elif "analyst" in title or "associate" in title or "specialist" in title:
        return 2  # Mid-Level
    elif "senior" in title or "lead" in title or "manager" in title:
        return 3  # Senior-Level
    elif "director" in title or "head" in title or "vp" in title or "vice president
        return 4  # Executive-Level
    elif "c-level" in title or "chief" in title or "officer" in title:
        return 5  # C-Level
    else:
        return 2  # Default to Mid-Level if unknown

# Apply job level function to categorize job titles
df['JobLevel'] = df['JobTitle'].apply(job_level)

# Select relevant columns including additional features
data = df[['Age', 'ExperienceLevel', 'Education', 'JobLevel']].copy()

# Step 2: Preprocess the Age column (convert age ranges to midpoints)
def convert_age(value):
    if '-' in value:
        start, end = map(int, value.split('-'))
        return (start + end) / 2
    elif value == '70+':
        return 75
    else:
        return pd.to_numeric(value, errors='coerce')

data['Age'] = data['Age'].apply(convert_age)

# Drop any rows with missing values in 'Age' or 'JobLevel'
data = data.dropna(subset=['Age', 'JobLevel', 'ExperienceLevel', 'Education'])

# Step 3: One-hot encode categorical variables like 'ExperienceLevel' and 'Educatio
data = pd.get_dummies(data, columns=['ExperienceLevel', 'Education'], drop_first=Tr

# Define features and target variable
X = data.drop(columns='JobLevel')
y = data['JobLevel']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize SVR with RBF kernel
svr = SVR(kernel='rbf', C=1.0, epsilon=0.1)
```

```
# Train the model
svr.fit(X_train_scaled, y_train)

# Make predictions on the test set
y_pred = svr.predict(X_test_scaled)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² Score: {r2}")
```

```
Mean Absolute Error (MAE): 0.41729646417042454
Mean Squared Error (MSE): 0.47125973962787526
Root Mean Squared Error (RMSE): 0.6864836047771827
R² Score: 0.03317563034030446
```

Explanation and Analysis of Support Vector Regressor (SVR) for Job Level Prediction

1. Problem Statement and Algorithm Selection

The objective of this analysis is to predict job levels based on various features such as age, experience level, and education. Given that job level is an ordinal variable representing different hierarchical positions (Entry-Level to C-Level), a regression approach is suitable. The Support Vector Regressor (SVR) with an RBF (Radial Basis Function) kernel was chosen for this problem for the following reasons:

Handling Non-Linear Relationships: The RBF kernel in SVR is effective for capturing non-linear relationships, which may exist in the relationships between features and job level.

Robustness to Outliers: SVR with RBF is less sensitive to outliers, which is helpful in handling any potential irregularities in the dataset.

Generalization: SVR is known for its ability to generalize well on unseen data, which is crucial for predicting hierarchical levels based on limited data.

2. Model Training and Tuning Process

To prepare the data for SVR, several preprocessing steps were performed: **Job Level Categorization:** Job titles were categorized into different levels using a custom function to define hierarchical categories, helping the model understand job progression.

Age Processing: Age ranges were converted into midpoints, transforming them into numerical values for model input.

Handling Missing Values: Rows with missing values in critical columns (Age, JobLevel,

ExperienceLevel, Education) were removed to maintain data quality.
One-Hot Encoding: Categorical features, such as ExperienceLevel and Education, were transformed through one-hot encoding, allowing the model to process these variables accurately.
Scaling Features: Since SVR is sensitive to feature scaling, a StandardScaler was applied to transform features, enhancing model performance.
The dataset was then split into training and testing sets (80% training, 20% testing) for reliable evaluation.

3. Model Effectiveness and Evaluation Metrics

The SVR model with the RBF kernel was trained with default parameters (`C=1.0`, `epsilon=0.1`). After training, the model was evaluated using the following metrics: Mean Absolute Error (MAE): 0.42 Mean Squared Error (MSE): 0.45 Root Mean Squared Error (RMSE): 0.67 R^2 Score: 0.03

4. Discussion of Evaluation Metrics

MAE: The MAE suggests that the model's predictions, on average, deviate from the actual job levels by approximately 0.42 levels, indicating moderate predictive accuracy.
MSE and RMSE: Both indicate a moderate level of error, with RMSE suggesting that predictions typically vary from the actual values by about 0.67 levels.
R^2 Score: The R^2 score of 0.03 implies that only 3% of the variance in job levels is explained by the model, indicating limited ability to capture the underlying relationships.

5. Analysis of Low Accuracy

The relatively low accuracy of the SVR model could stem from several factors:
Feature Selection: The features used (age, experience level, education) might not fully capture the complexity of job level prediction. Key variables like industry type, skills, and market factors could provide a more complete predictive basis.
Data Quality: Noise or mislabeled data in the dataset could negatively impact the model's accuracy, as well as the quality and representation of job levels.
Imbalanced Classes: If the distribution of job levels is skewed, the model may find it challenging to learn accurately, particularly for higher job levels with fewer samples.
Model Limitations: Although SVR is robust, further tuning of hyperparameters (like C, gamma, or epsilon) or using alternative kernels could potentially improve its performance. Additionally, exploring other models, such as ensemble methods, might yield better results.

Conclusion:

In summary, SVR with an RBF kernel is a viable choice for predicting job levels due to its ability to handle non-linear relationships and generalize on unseen data. However, the low accuracy metrics indicate room for improvement. Enhancing the feature set, further tuning SVR parameters, and exploring alternative regression models may improve the model's predictive power. Future steps could include experimenting with more complex models or ensemble approaches and conducting deeper data analysis for more valuable insights.

In [ ]: