Predict Vehicle Prices using Vehicle dataset

Background: The purpose of this project is to develop a machine learning model that predicts the price of a vehicle based on several key attributes such as make, model, mileage, age, and more. You will use a variety of regression models, including Random Forest, Gradient Boosting, Multi-layer Perceptron (MLP), and Linear Regression, to predict vehicle prices. The models will be evaluated based on their performance, and you will also perform feature importance analysis to understand which factors have the most influence on vehicle price predictions.

The project involves tasks such as data preprocessing, model training, hyperparameter tuning, model evaluation, and feature analysis. The goal is to build an accurate predictive model while gaining valuable insights into the data.

Project Tasks:

1. Data Preprocessing:

- **Loading the Dataset:** The dataset will be loaded from a CSV file using pandas.

```
df = pd.read_csv("dataset.csv")
```

- **Handling Missing Data:** Impute missing values in the dataset. For example, the *price* column can be imputed with the median, and other categorical columns like *make*, *model*, and *fuel* can be imputed using the most frequent value.

```
imputer = SimpleImputer(strategy='median')
df['price'] = imputer.fit_transform(df[['price']])
```

- **Feature Engineering:** Add a new feature *vehicle_age* by calculating the difference between the current year and the year of manufacture. Drop irrelevant columns such as *name*, *description*, and *year*.

```
current_year = datetime.datetime.now().year
df['vehicle_age'] = current_year - df['year']
df.drop(columns=['name', 'description', 'year'], inplace=True)
```

- **Visualizing Missing Values and Data Distribution:** Use heatmaps to check for missing values and histograms to understand the price distribution.

```
plt.figure(figsize=(10, 5))
sns.heatmap(df.isnull(), cmap='viridis', cbar=False, yticklabels=False)
plt.title("Missing Values Heatmap")
plt.show()

plt.figure(figsize=(8, 5))
sns.histplot(df['price'], bins=50, kde=True)
plt.title("Price Distribution")
plt.xlabel("Price (USD)")
plt.ylabel("Frequency")
plt.show()
```

2. Feature Engineering:

- **Categorical and Numerical Features:** Identify the numerical and categorical features in the dataset. For example:

```
categorical_features = ['make', 'model', 'engine', 'fuel', 'transmission',
                        'trim', 'body', 'drivetrain', 'exterior_color', 'interior_color']
numerical_features = ['mileage', 'cylinders', 'doors', 'vehicle_age']
```

- **Pipeline for Preprocessing:** Build separate pipelines for numerical and categorical features, where numerical features are imputed and scaled, and categorical features are imputed and one-hot encoded.

```
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

cat_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer([
    ('num', num_pipeline, numerical_features),
    ('cat', cat_pipeline, categorical_features)
])
```

3. Model Training:

- **Splitting the Data:** Split the dataset into training and testing sets for model evaluation.

```
X = df.drop(columns=['price'])
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **Model Selection and Hyperparameter Tuning:** Train multiple regression models and use Grid Search to optimize the hyperparameters. For example:

```
param_grid = {
    'RandomForestRegressor': {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 10, 20]
    },
    'GradientBoostingRegressor': {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.1, 0.2]
    },
    'MLPRegressor': {
        'hidden_layer_sizes': [(64, 32), (128, 64)],
        'max_iter': [500, 1000]
    }
}
```

- **Fitting the Model:** For each model, apply the best hyperparameters found through grid search and evaluate the performance.

```
for model_name, model in models.items():
    if model_name in param_grid:
        grid_search = GridSearchCV(model, param_grid[model_name], cv=3, scoring='r2', n_jobs=-1)
        grid_search.fit(X_train_processed, y_train)
        best_models[model_name] = grid_search.best_estimator_
    else:
        model.fit(X_train_processed, y_train)
        best_models[model_name] = model
```

4. Model Evaluation:

- **Evaluation Metrics:** Evaluate each model using performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score.

```python
def evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"{model.__class__.__name__} Performance:")
    print("MAE:", mean_absolute_error(y_test, y_pred))
    print("MSE:", mean_squared_error(y_test, y_pred))
    print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
    print("R² Score:", r2_score(y_test, y_pred))
    print("-----------------------------")
    plt.figure(figsize=(8, 5))
    sns.scatterplot(x=y_test, y=y_pred, alpha=0.4)
    sns.regplot(x=y_test, y=y_pred, scatter=False, color='red')
    plt.xlabel("Actual Prices")
    plt.ylabel("Predicted Prices")
    plt.title(f"Actual vs Predicted Prices - {model.__class__.__name__}")
    plt.show()
```

5. Feature Importance Analysis:

- **Feature Importance Calculation:** After training the Random Forest model, extract the feature importance scores and visualize them.

```python
rf = best_models['RandomForestRegressor']
feature_importance = rf.feature_importances_
feature_names = preprocessor.get_feature_names_out()

sorted_indices = np.argsort(feature_importance)[::-1]
feature_importance = feature_importance[sorted_indices]
feature_names = feature_names[sorted_indices]

plt.figure(figsize=(14, 8))
sns.barplot(x=feature_importance, y=feature_names, palette="viridis")
plt.xlabel("Feature Importance", fontsize=12)
plt.ylabel("Feature", fontsize=12)
plt.title("Random Forest Feature Importance", fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```

6. Visualizing Top Features:

- **Top 20 Features:** Visualize the top 20 most important features that affect the vehicle price.

```python
num_features = 20  # Adjust as needed
plt.figure(figsize=(12, 6))
sns.barplot(x=feature_importance[:num_features], y=feature_names[:num_features], palette="viridis")
plt.xlabel("Feature Importance", fontsize=12)
plt.ylabel("Feature", fontsize=12)
plt.title("Top 20 Most Important Features", fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```

Deliverables:

1. **Python Code:**
   - Complete code for data preprocessing, model training, and evaluation.
2. **Visualizations:**
   - Visualizations for missing data, price distribution, model performance (Actual vs Predicted), and feature importance.
3. **Report:**
   - A final report that includes:
     - Comparison of model performance using various metrics (MAE, MSE, RMSE, R^2 Score).
     - Insights from feature importance analysis, highlighting the most significant factors affecting vehicle price.
4. **Final Predictions:**
   - The best-performing model and its predictions on the test dataset.

Objective:

The aim of this project is to predict vehicle prices using machine learning models, evaluate their effectiveness, and gain actionable insights from the feature importance analysis. This information will be valuable for making informed decisions about vehicle pricing and understanding the key factors that drive price variations.