Title: Standard Machine Learning Project Steps (House Price Prediction Example)

Standard Machine Learning Project Steps

This document outlines all the essential steps required to create a professional machine learning project, using house price prediction as an example.

---

1. Problem Definition

- Define the problem clearly.
- Example: Predict house prices based on property features.
- Type: Regression problem (continuous output).
- Goal: Minimize RMSE and maximize R^2.

Resume/Kaggle Note:

> Built a predictive model to estimate house prices using structured tabular data.

---

2. Data Collection / Loading

- Load data from CSV, database, or API.
- Example:

```
import pandas as pd
data = pd.read_csv('/kaggle/input/housedata/data.csv')
print(data.head())
```

---

3. Data Understanding

- Check dataset structure, types, missing values, and basic statistics.
- Example:

```
print(data.info())
print(data.describe())
print(data.isnull().sum())
```

Resume Note:

Performed initial data exploration and understanding of dataset features.

---

4. Exploratory Data Analysis (EDA)

- Visualize data and relationships to understand patterns.
- Must include:
- Distributions (histograms)
- Outliers (boxplots)
- Relationships (scatter plots)
- Correlations (heatmaps)
- Categorical analysis (bar plots)

Example (Matplotlib-only):

```
import matplotlib.pyplot as plt
plt.hist(data['price'], bins=50)
plt.title("Price Distribution")
plt.show()
```

Resume Note:

Performed EDA including feature distributions, correlations, and outlier detection.

---

5. Data Cleaning

- Fix missing values, remove irrelevant columns, handle duplicates.
- Example:

```
data = data.drop(['street','city','date'], axis=1)
data = data.fillna(data.median())
```

---

6. Feature Engineering

- Create new features or transform existing ones.
- Example:

```
data['house_age'] = 2025 - data['yr_built']
data['price_per_sqft'] = data['price'] / data['sqft_living']
```

---

7. Feature Selection

- Decide which features to include.
- Example:

```
X = data.drop(['price','yr_built'], axis=1)
Y = data['price']
```

---

8. Train-Test Split

- Split data into training and testing sets.
- Example:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
```

---

9. Feature Scaling (Optional)

- Scale features for models that need it (LR, KNN, SVR).
- Example:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

---

10. Model Selection

- Choose appropriate models.
- Example (Regression):

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor

models = {
    'Linear Regression': LinearRegression(),
```

```
    'Random Forest': RandomForestRegressor(),
    'Decision Tree': DecisionTreeRegressor(),
    'KNN': KNeighborsRegressor()
}
```

---

11. Model Training

- Fit each model on training data.
- Example:

```
for name, model in models.items():
    model.fit(X_train_scaled, Y_train)
```

---

12. Model Evaluation

- Evaluate models using MSE, RMSE, MAE, R^2.
- Example:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
for name, model in models.items():
    Y_pred = model.predict(X_test_scaled)
    print(name)
    print("RMSE:", mean_squared_error(Y_test,Y_pred,squared=False))
    print("MAE:", mean_absolute_error(Y_test,Y_pred))
    print("R2:", r2_score(Y_test,Y_pred))
```

---

13. Model Comparison

- Compare metrics to select the best model.
- Example Table: | Model | MSE | RMSE | MAE | R^2 | |-------|-----|------|-----|----| | LR | ... | ... | ... | ... | | RF | ... | ... | ... | ... | | DT | ... | ... | ... | ... | | KNN | ... | ... | ... | ... |

Resume Note:

> Random Forest achieved best performance with RMSE = 150k and R^2 = 0.85.

---

14. Model Interpretation (Optional)

- Explain model behavior using feature importance or SHAP values.
- Example:

```
importances = models['Random Forest'].feature_importances_
plt.barh(X.columns, importances)
plt.title("Feature Importance")
plt.show()
```

---

15. Hyperparameter Tuning (Optional)

- Example:

```
from sklearn.model_selection import GridSearchCV
param_grid = {'n_estimators':[50,100,200],'max_depth':[10,20,None]}
grid = GridSearchCV(RandomForestRegressor(), param_grid, cv=5)
grid.fit(X_train_scaled, Y_train)
print(grid.best_params_)
```

---

16. Final Model Selection

- Select the model with best performance.
- Resume Note: Final model: Random Forest Regressor, RMSE = 145k, R^2 = 0.87

---

17. Deployment / Reporting (Optional)

- Streamlit / Flask / Save with pickle.
- Example:

```
import pickle
with open('house_price_model.pkl','wb') as f:
    pickle.dump(models['Random Forest'], f)
```

---

18. Documentation & Presentation

- Include markdown explanations, plots, and a final summary.

• Upload notebook to Kaggle or GitHub.

Resume Example:

> Built a complete house price prediction pipeline including EDA, preprocessing, feature engineering, model training and evaluation, achieving RMSE 145k and R^2 0.87.

✅Summary Table of Steps

Step	Description
1	Problem Definition
2	Data Loading
3	Data Understanding
4	EDA (Plots, correlations, outliers)
5	Data Cleaning
6	Feature Engineering
7	Feature Selection
8	Train-Test Split
9	Feature Scaling
10	Model Selection
11	Model Training
12	Model Evaluation
13	Model Comparison
14	Model Interpretation
15	Hyperparameter Tuning
16	Final Model Selection
17	Deployment / Reporting
18	Documentation & Presentation