Cheatsheet: Model Evaluation and Refinement - Used Cars Pricing

Overview

This cheatsheet provides a quick reference for evaluating and refining a model for used cars pricing. It includes key concepts, evaluation metrics, and practical tips for model refinement.

Key Concepts

1. Model Evaluation

- **Accuracy**: Measures the proportion of correct predictions.
- **Precision**: Measures the proportion of true positive predictions among all positive predictions.
- **Recall**: Measures the proportion of true positive predictions among all actual positives.
- **F1 Score**: Harmonic mean of precision and recall.
- **Mean Absolute Error (MAE)**: Average of absolute errors between predicted and actual values.
- **Mean Squared Error (MSE)**: Average of squared errors between predicted and actual values.
- **R-squared (R^2)**: Measures how well the model explains the variance in the data.

2. Cross-Validation

- **K-Fold Cross-Validation**: Splits the data into k subsets and trains the model on k-1 subsets while testing on the remaining subset. This is repeated k times.

- **Stratified K-Fold**: Ensures that each fold has a representative distribution of the target variable.

3. Model Refinement

- **Hyperparameter Tuning**: Adjusts the parameters of the model to improve performance.
- **Feature Selection**: Selects the most relevant features to improve model performance and reduce overfitting.
- **Regularization**: Adds a penalty to the loss function to prevent overfitting (e.g., L1 and L2 regularization).
- **Ensemble Methods**: Combines multiple models to improve performance (e.g., Random Forest, Gradient Boosting).

Practical Tips

1. Evaluate Model Performance

- Use cross-validation to assess model performance on unseen data.
- Compare different evaluation metrics to get a comprehensive understanding of model performance.

2. Refine the Model

- Use grid search or random search for hyperparameter tuning.
- Select features based on their importance and relevance to the target variable.
- Apply regularization techniques to prevent overfitting.
- Consider using ensemble methods to improve model accuracy and robustness.

3. Visualize Results

- Use plots to visualize model performance (e.g., residual plots, feature importance plots).
- Use confusion matrices for classification tasks and residual plots for regression tasks.

Tools and Libraries

- **Pandas**: For data manipulation and analysis.
- **Scikit-learn**: For model evaluation, cross-validation, and hyperparameter tuning.
- **Matplotlib and Seaborn**: For data visualization.
- **XGBoost and LightGBM**: For advanced ensemble methods.

References

- Scikit-learn Documentation
- XGBoost Documentation
- LightGBM Documentation

Conclusion

Model evaluation and refinement are crucial steps in building an accurate and robust model for used cars pricing. By understanding the evaluation metrics and applying practical refinement techniques, you can improve the performance of your model and make more accurate predictions.

Syntax Codes for Model Evaluation and Refinement

1. Importing Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2
_score, classification_report, confusion_matrix
from sklearn.ensemble import RandomForestRegressor, GradientBoostingReg
ressor
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Splitting Data

```
X = df.drop('price', axis=1)
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 random_state=42)
```

3. Training a Model

```
model = LinearRegression()
model.fit(X_train, y_train)
```

4. Making Predictions

```
y_pred = model.predict(X_test)
```

5. Evaluating Model Performance

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"MAE: {mae}, MSE: {mse}, R²: {r2}")
```

6. Cross-Validation

```
scores = cross_val_score(model, X, y, cv=5)
print(f"Cross-Validation Scores: {scores}")
```

7. Hyperparameter Tuning with Grid Search

```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5]
}
grid_search = GridSearchCV(RandomForestRegressor(), param_grid, cv=5)
grid_search.fit(X_train, y_train)
print(f"Best Parameters: {grid_search.best_params_}")
```

8. Feature Selection

```
selector = SelectKBest(score_func=f_regression, k=5)
X_selected = selector.fit_transform(X, y)
```

9. Regularization (Lasso Regression)

```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
```

10. Ensemble Methods (Random Forest)

```
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

11. Visualizing Feature Importance

```
importances = rf.feature_importances_
sns.barplot(x=importances, y=X.columns)
plt.title("Feature Importance")
plt.show()
```

12. Residual Plot

```
residuals = y_test - y_pred
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.title("Residual Plot")
plt.show()
```

13. Confusion Matrix (Classification Example)

```
from sklearn.metrics import confusion_matrix
y_pred_class = np.round(y_pred)
cm = confusion_matrix(y_test, y_pred_class)
sns.heatmap(cm, annot=True, fmt='d')
plt.title("Confusion Matrix")
plt.show()
```

14. Pipeline for Preprocessing and Modeling

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LinearRegression())
])
pipeline.fit(X_train, y_train)
y_pred_pipeline = pipeline.predict(X_test)
```

15. Randomized Search for Hyperparameter Tuning

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint, uniform


param_dist = {
    'n_estimators': randint(100, 300),
    'max_depth': [None, 5, 10],
    'learning_rate': uniform(0.01, 0.2)
}


random_search = RandomizedSearchCV(GradientBoostingRegressor(), param_d
ist, n_iter=10, cv=5)
random_search.fit(X_train, y_train)
print(f"Best Parameters: {random_search.best_params_}")
```

16. Saving the Model

```
import joblib
joblib.dump(model, 'model.pkl')
```

17. Loading the Model

```
model = joblib.load('model.pkl')
```

18. Plotting Regression Line

```
sns.scatterplot(x=y_test, y=y_pred)
sns.lineplot(x=y_test, y=y_pred, color='red')
plt.title("Actual vs Predicted Values")
plt.show()
```

19. Plotting Distribution of Residuals

```
sns.histplot(residuals, kde=True)
plt.title("Distribution of Residuals")
plt.show()
```

20. Plotting Feature Importance for Random Forest

```
importances = rf.feature_importances_
sns.barplot(x=importances, y=X.columns)
plt.title("Feature Importance")
plt.show()
```

Conclusion

These syntax codes provide a practical guide to understanding how the code works for model evaluation and refinement in used cars pricing. They cover data splitting, model training, evaluation metrics, cross-validation, hyperparameter tuning, feature selection, regularization, ensemble methods, and visualization techniques.