Automated model Ensemble Techniques for Improved Accuracy

Phase 3: Model Training and Evaluation

3.1 Overview of Model Training and Evaluation

Training and evaluating automated model ensemble techniques for improved accuracy involve several steps that work together to create a system where multiple models collaborate to provide better performance than individual models. The key goal is to improve predictive accuracy by combining the strengths of multiple models, while mitigating their weaknesses.

3.2 Choosing Suitable Algorithms

For the **Automated model ensemble techniques for improved accuracy** project, the key algorithms are:

1. Bagging (Bootstrap Aggregating)

Algorithm: Random Forest

How it Works: Multiple models (e.g., decision trees) are trained on bootstrapped datasets, and their predictions are averaged (regression) or voted (classification).

Key Benefit: Reduces variance and prevents overfitting.

2. Boosting

Algorithms: AdaBoost, Gradient Boosting (GBM), XGBoost, LightGBM, CatBoost

How it Works: Models are trained sequentially, with each model focusing on correcting the errors of the previous ones.

Key Benefit: Reduces bias and improves accuracy, especially for complex data.

Source code:

```
sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd

# Load dataset
```

```
data = pd.read_csv('data.csv')
X = data.drop('target', axis=1)
y = data['target']

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the model
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd

# Load dataset
data = pd.read_csv('data.csv')
X = data.drop('target', axis=1)
y = data['target']

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the model
base_model = DecisionTreeClassifier(max_depth=1)
model = AdaBoostClassifier(base_estimator=base_model, n_estimators=50)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

3.3 Automating Hyperparameter Tuning

Using optuna for **Bayesian Optimization**

```
import optuna

from sklearn.ensemble import RandomForestRegressor

from sklearn.model_selection import cross_val_score



def objective(trial):

    n_estimators = trial.suggest_int('n_estimators', 50, 500)

    max_depth = trial.suggest_int('max_depth', 3, 20)

    min_samples_split = trial.suggest_int('min_samples_split', 2, 10)

    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 10)



    model = RandomForestRegressor(n_estimators=n_estimators, max_depth=max_depth,

                                  min_samples_split=min_samples_split,
min_samples_leaf=min_samples_leaf)

    score = cross_val_score(model, X_train, y_train, cv=5,
scoring='neg_mean_absolute_error').mean()

    return -score



study = optuna.create_study(direction='minimize')

study.optimize(objective, n_trials=50)

print(study. best_params)
```

3.4 Model Evaluation Metrics

1.Mean Absolute Error

```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test, y_pred)

print("MAE:", mae)
```

2.Mean Squared Error(MES)

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, y_pred)

print("MSE:", mse)
```

3.Root Mean Squared Error(RMSE)

```
rmse = mean_squared_error(y_test, y_pred, squared=False)

print("RMSE:", rmse)
```

3.5 Cross-Validation

Use k-fold cross-validation to ensure generalizability.

Source code:

```
from sklearn.model_selection import KFold,
cross_val_score
from sklearn.ensemble import
RandomForestRegressor
from sklearn.metrics import
mean_absolute_error, make_scorer

# Define K-Fold Cross Validation
kf = KFold(n_splits=5, shuffle=True,
random_state=42)

# Define model
```

```
model =
RandomForestRegressor(n_estimators=100,
random_state=42)

# Use Negative MAE for scoring
mae_scorer = make_scorer(mean_absolute_error,
greater_is_better=False)

# Perform Cross-Validation
cv_scores = cross_val_score(model, X, y, cv=kf,
scoring=mae_scorer)

# Print Results
print("Cross-Validation MAE Scores:", -
cv_scores)
print("Mean MAE:", -cv_scores.mean())
```

3.6 Conclusion of Phase 3

By combining ensemble learning, hyperparameter tuning, and K-Fold Cross-Validation, we can significantly improve the accuracy and reliability of house price prediction models. Use XGBoost/LightGBM with Optuna for tuning,Validate models using K-Fold CV and MAE/RMSE,Test stacking ensembles for further accuracy gains.