```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Load data
data = pd.read_csv('house_prices.csv')  # Load your data file (e.g., 'house_prices.csv')

# Data preprocessing
# Identify features and target variable
features = data.drop(columns=['price'])  # Replace 'price' with your target variable column name
target = data['price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=42)

# Identify numerical and categorical features
numerical_features = features.select_dtypes(include=np.number).columns
categorical_features = features.select_dtypes(include=['object']).columns

# Preprocess numerical and categorical features
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
```

```
# Random Forest model for house price prediction
rf_model = Pipeline([
    ('preprocessor', preprocessor),
    ('model', RandomForestRegressor(random_state=42))
])

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

# Predict house prices on the test set
y_pred_rf = rf_model.predict(X_test)

# Evaluate the Random Forest model
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)

print(f"Random Forest - Mean Squared Error: {mse_rf}")
print(f"Random Forest - Mean Absolute Error: {mae_rf}")

# TensorFlow neural network model for house price prediction
nn_model = Sequential([
    Dense(64, activation='relu', input_shape=(len(X_train.columns),)),
    Dense(64, activation='relu'),
    Dense(1)
])

# Compile the model
nn_model.compile(optimizer='adam', loss='mse')

# Preprocess the data
X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)

# Train the neural network model
nn_model.fit(X_train_preprocessed, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Predict house prices on the test set
y_pred_nn = nn_model.predict(X_test_preprocessed)

# Evaluate the neural network model
mse_nn = mean_squared_error(y_test, y_pred_nn)
mae_nn = mean_absolute_error(y_test, y_pred_nn)

print(f"Neural Network - Mean Squared Error: {mse_nn}")
print(f"Neural Network - Mean Absolute Error: {mae_nn}")

# Choose the model with the best performance based on evaluation metrics

# Deploy the best model for predictions
best_model = rf_model if mse_rf < mse_nn else nn_model

# Make predictions on new data
new_data = pd.DataFrame({'feature1': [value1], 'feature2': [value2], ...})  # Replace with new data
new_data_preprocessed = preprocessor.transform(new_data)
predicted_price = best_model.predict(new_data_preprocessed)

print(f"Predicted House Price: {predicted_price}")
```