import streamlit as st  
import pandas as pd  
import numpy as np  
import joblib  
import pickle  
from datetime import datetime  
  
st.set\_page\_config(  
 page\_title="Недвижимость",  
 page\_icon="🏢",  
 layout="wide",  
 initial\_sidebar\_state="expanded"  
)  
  
st.markdown("""  
<style>  
.main-header {  
 font-size: 2.5rem;  
 color: #1f77b4;  
 text-align: center;  
 margin-bottom: 1rem;  
}  
.prediction-card {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 padding: 2rem;  
 border-radius: 15px;  
 color: white;  
 text-align: center;  
 margin: 1rem 0;  
 box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);  
}  
.price-display {  
 font-size: 2.5rem;  
 font-weight: bold;  
 color: #ffd700;  
 margin: 1rem 0;  
}  
.type-display {  
 font-size: 1.8rem;  
 font-weight: bold;  
 color: #00ff00;  
 margin: 0.5rem 0;  
}  
.feature-card {  
 background: #f8f9fa;  
 padding: 1rem;  
 border-radius: 10px;  
 margin: 0.5rem 0;  
 border-left: 4px solid #1f77b4;  
}  
.stButton>button {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 border: none;  
 padding: 0.5rem 2rem;  
 border-radius: 25px;  
 font-weight: bold;  
}  
.info-box {  
 background: #e8f4fd;  
 padding: 1rem;  
 border-radius: 10px;  
 margin: 1rem 0;  
 border-left: 4px solid #2196F3;  
}  
.currency-selector {  
 background: #f0f8ff;  
 padding: 1rem;  
 border-radius: 10px;  
 margin: 1rem 0;  
 border: 2px solid #1e90ff;  
}  
.history-card {  
 background: #fff3cd;  
 padding: 1rem;  
 border-radius: 10px;  
 margin: 0.5rem 0;  
 border-left: 4px solid #ffc107;  
}  
</style>  
""", unsafe\_allow\_html=True)  
  
EXCHANGE\_RATES = {  
 'TRY': 4.0, # 1 лира = 4 рубля (примерный курс)  
 'USD': 90.0, # 1 доллар = 90 рублей  
 'EUR': 100.0, # 1 евро = 100 рублей  
 'RUB': 1.0 # 1 рубль = 1 рубль  
}  
  
  
@st.cache\_resource  
def load\_models\_cached():  
 models = {}  
 try:  
 models['type\_classifier'] = joblib.load('property\_type\_classifier.pkl')  
 models['price\_model'] = joblib.load('stacking\_price\_predictor.pkl')  
  
 with open('feature\_encoders.pkl', 'rb') as f:  
 models['feature\_encoders'] = pickle.load(f)  
 with open('label\_encoder.pkl', 'rb') as f:  
 models['type\_encoder'] = pickle.load(f)  
 with open('type\_scaler.pkl', 'rb') as f:  
 models['type\_scaler'] = pickle.load(f)  
 with open('scaler.pkl', 'rb') as f:  
 models['price\_scaler'] = pickle.load(f)  
  
 return models  
  
 except Exception as e:  
 st.error(f"Ошибка загрузки: {str(e)}")  
 return None  
  
  
def convert\_to\_rubles(price, currency):  
 return price \* EXCHANGE\_RATES.get(currency, 1.0)  
  
  
def format\_price(price, currency='RUB'):  
 if currency == 'RUB':  
 return f"{price:,.0f} ₽"  
 elif currency == 'USD':  
 return f"${price:,.0f}"  
 elif currency == 'EUR':  
 return f"€{price:,.0f}"  
 elif currency == 'TRY':  
 return f"{price:,.0f} ₺"  
 else:  
 return f"{price:,.0f}"  
  
  
def main():  
 if 'models' not in st.session\_state:  
 st.session\_state.models = load\_models\_cached()  
 if 'predictions\_history' not in st.session\_state:  
 st.session\_state.predictions\_history = []  
 if 'current\_prediction' not in st.session\_state:  
 st.session\_state.current\_prediction = None  
  
 models = st.session\_state.models  
  
 st.markdown('<div class="main-header">🏢Оценка недвижимости</div>', unsafe\_allow\_html=True)  
 st.markdown("### Введите параметры объекта для точной оценки стоимости и определения типа")  
  
 st.markdown('<div class="currency-selector">', unsafe\_allow\_html=True)  
 display\_currency = st.selectbox(  
 "Валюта для отображения результата",  
 ["RUB", "USD", "EUR", "TRY"],  
 format\_func=lambda x: {  
 "RUB": "Рубли (₽)",  
 "USD": "Доллары ($)",  
 "EUR": "Евро (€)",  
 "TRY": "Турецкие лиры (₺)"  
 }[x],  
 help="Выберите валюту для отображения цены"  
 )  
 st.markdown('</div>', unsafe\_allow\_html=True)  
  
 with st.form("property\_form"):  
 col1, col2 = st.columns(2)  
  
 with col1:  
 st.markdown("### 📋 Основные параметры")  
  
 property\_type = st.selectbox(  
 "Тип недвижимости",  
 ["Yazlık", "Villa", "Loft", "Daire"],  
 help="Основной тип недвижимости"  
 )  
  
 city = st.selectbox(  
 "Город",  
 ["İstanbul", "Ankara", "İzmir", "Muğla", "Antalya", "Bursa", "Mersin"],  
 help="Выберите город расположения"  
 )  
  
 district = st.text\_input(  
 "Район",  
 "Kartal",  
 help="Введите название района"  
 )  
  
 neighborhood = st.text\_input(  
 "Микрорайон",  
 "Levent",  
 help="Введите название микрорайона")  
  
 with col2:  
 st.markdown("### 📏 Технические характеристики")  
  
 size = st.slider(  
 "Площадь (m²)",  
 min\_value=10,  
 max\_value=500,  
 value=90,  
 help="Общая площадь объекта"  
 )  
  
 room\_count = st.slider(  
 "Количество комнат",  
 min\_value=1,  
 max\_value=10,  
 value=3,  
 help="Общее количество комнат"  
 )  
  
 building\_age = st.slider(  
 "Возраст здания (лет)",  
 min\_value=0,  
 max\_value=50,  
 value=5,  
 help="Сколько лет зданию"  
 )  
  
 st.markdown("### ⚙️ Дополнительные параметры")  
 col3, col4 = st.columns(2)  
  
 with col3:  
 total\_floor\_count = st.slider(  
 "Этажность дома",  
 min\_value=1,  
 max\_value=30,  
 value=10,  
 help="Общее количество этажей в здании"  
 )  
  
 floor\_no = st.slider(  
 "Этаж расположения",  
 min\_value=0,  
 max\_value=30,  
 value=5,  
 help="На каком этаже расположен объект"  
 )  
  
 with col4:  
 price\_currency = st.selectbox(  
 "Валюта объекта",  
 ["TRY", "USD", "EUR"],  
 format\_func=lambda x: {  
 "TRY": "Турецкая лира (₺)",  
 "USD": "Доллар ($)",  
 "EUR": "Евро (€)"  
 }[x],  
 help="В какой валюте оценивается объект"  
 )  
  
 heating\_type = st.selectbox(  
 "Тип отопления",  
 ["Fancoil", "Merkezi", "Doğalgaz", "Klima", "Soba", "Yok"],  
 help="Тип системы отопления"  
 )  
  
 listing\_type = st.radio(  
 "Тип сделки",  
 [1, 2],  
 format\_func=lambda x: "Продажа" if x == 1 else "Аренда",  
 help="Продажа или аренда объекта"  
 )  
  
 time\_on\_market = st.slider(  
 "Время на рынке (дни)",  
 min\_value=0,  
 max\_value=365,  
 value=30,  
 help="Сколько дней объект на рынке"  
 )  
  
 submitted = st.form\_submit\_button("🎯 Рассчитать стоимость и тип", use\_container\_width=True)  
  
 if submitted:  
 try:  
 demo\_mode = True  
 predicted\_type = "Daire"  
 predicted\_price\_rub = 0  
  
 st.markdown("### 🔍 Определение типа недвижимости")  
  
 if models and models.get('type\_classifier') and models.get('type\_scaler'):  
 try:  
 type\_features = pd.DataFrame({  
 'type': [encode\_property\_type(property\_type)], # ИСПРАВЛЕНО  
 'listing\_type': [listing\_type],  
 'tom': [time\_on\_market],  
 'building\_age': [building\_age],  
 'total\_floor\_count': [total\_floor\_count],  
 'floor\_no': [floor\_no],  
 'room\_count': [room\_count],  
 'size': [size],  
 'city': [encode\_city(city)],  
 'district': [encode\_district(district)],  
 'neighborhood': [encode\_neighborhood(neighborhood)],  
 'heating\_type': [encode\_heating\_type(heating\_type)],  
 'price\_currency': [0],  
 'start\_month': [datetime.now().month]  
 })  
  
 st.sidebar.write("📊 Признаки для типа:", type\_features.iloc[0].to\_dict())  
  
 scaled\_type\_features = models['type\_scaler'].transform(type\_features)  
  
 type\_prediction = models['type\_classifier'].predict(scaled\_type\_features)[0]  
  
 if models.get('type\_encoder'):  
 predicted\_type = models['type\_encoder'].inverse\_transform([type\_prediction])[0]  
 else:  
 predicted\_type = decode\_property\_type(type\_prediction)  
  
 demo\_mode = False  
  
 st.sidebar.write("🎯 Предсказание:", type\_prediction)  
 st.sidebar.write("🏠 Декодированный тип:", predicted\_type)  
  
 except Exception as e:  
 st.error(f"❌ Ошибка предсказания типа: {str(e)}")  
 predicted\_type = "Daire"  
 st.sidebar.error(f"Ошибка: {e}")  
  
 if models and models.get('price\_model') and models.get('price\_scaler'):  
 try:  
 price\_features = pd.DataFrame({  
 'type': [encode\_property\_type(property\_type)],  
 'sub\_type': [encode\_property\_type(predicted\_type)],  
 'listing\_type': [int(listing\_type)],  
 'tom': [int(time\_on\_market)],  
 'building\_age': [int(building\_age)],  
 'total\_floor\_count': [float(total\_floor\_count)],  
 'floor\_no': [float(floor\_no)],  
 'room\_count': [int(room\_count)],  
 'size': [float(size)],  
 'city': [int(encode\_city(city))],  
 'district': [int(encode\_district(district))],  
 'neighborhood': [int(encode\_neighborhood(neighborhood))],  
 'heating\_type': [int(encode\_heating\_type(heating\_type))],  
 'price\_currency': [0],  
 'start\_month': [np.int32(datetime.now().month)]  
 })  
  
 expected\_features = models['price\_model'].feature\_names\_in\_  
 price\_features = price\_features[expected\_features]  
  
 scaled\_price\_features = models['price\_scaler'].transform(price\_features)  
  
 prediction = models['price\_model'].predict(scaled\_price\_features)[0]  
  
 if prediction < 20:  
 predicted\_price\_original = np.expm1(prediction)  
 else:  
 predicted\_price\_original = prediction  
  
 predicted\_price\_rub = convert\_to\_rubles(predicted\_price\_original, price\_currency)  
  
 except Exception as e:  
 st.error(f"❌ Ошибка предсказания цены: {str(e)}")  
 predicted\_price\_rub = calculate\_demo\_price(size, room\_count, building\_age, city, district)  
 demo\_mode = True  
 else:  
 predicted\_price\_rub = calculate\_demo\_price(size, room\_count, building\_age, city, district)  
 demo\_mode = True  
  
 current\_prediction = {  
 'type': predicted\_type,  
 'price\_rub': predicted\_price\_rub,  
 'price\_currency': price\_currency,  
 'display\_currency': display\_currency,  
 'demo': demo\_mode,  
 'size': size,  
 'room\_count': room\_count,  
 'city': city,  
 'district': district,  
 'timestamp': datetime.now().strftime("%H:%M:%S")  
 }  
  
 st.session\_state.current\_prediction = current\_prediction  
 st.session\_state.predictions\_history.append(current\_prediction)  
  
 if len(st.session\_state.predictions\_history) > 10:  
 st.session\_state.predictions\_history = st.session\_state.predictions\_history[-10:]  
  
 except Exception as e:  
 st.error(f"❌ Ошибка при расчете: {str(e)}")  
  
 if st.session\_state.current\_prediction:  
 prediction = st.session\_state.current\_prediction  
  
 st.markdown(f"""  
 <div class="prediction-card">  
 <h2>🏠 Предсказанный тип недвижимости</h2>  
 <div class="type-display">{prediction['type']}</div>  
 </div>  
 """, unsafe\_allow\_html=True)  
  
 display\_price = prediction['price\_rub'] / EXCHANGE\_RATES[display\_currency] if display\_currency != 'RUB' else \  
 prediction['price\_rub']  
  
 st.markdown(f"""  
 <div class="prediction-card">  
 <h2>💰 {'Предсказанная' if not prediction['demo'] else 'Примерная'} стоимость</h2>  
 <div class="price-display">{format\_price(display\_price, display\_currency)}</div>  
 <p>{'На основе ML модели' if not prediction['demo'] else 'Демо-режим'} • {prediction['timestamp']}</p>  
 </div>  
 """, unsafe\_allow\_html=True)  
  
 if st.session\_state.predictions\_history:  
 st.markdown("### 📋 История предсказаний")  
  
 for i, hist\_pred in enumerate(reversed(st.session\_state.predictions\_history[-5:]), 1):  
 display\_price = hist\_pred['price\_rub'] / EXCHANGE\_RATES[display\_currency] if display\_currency != 'RUB' else \  
 hist\_pred['price\_rub']  
  
 st.markdown(f"""  
 <div class="history-card">  
 <strong>#{len(st.session\_state.predictions\_history) - i + 1}</strong> • {hist\_pred['timestamp']}  
 <div>Тип: {hist\_pred['type']}</div>  
 <div>Стоимость: {format\_price(display\_price, display\_currency)}</div>  
 <div>Площадь: {hist\_pred['size']}m² • Комнат: {hist\_pred['room\_count']}</div>  
 <div>Город: {hist\_pred['city']} • Район: {hist\_pred['district']}</div>  
 </div>  
 """, unsafe\_allow\_html=True)  
  
 if st.button("🧹 Очистить историю"):  
 st.session\_state.predictions\_history = []  
 st.session\_state.current\_prediction = None  
 st.rerun()  
  
  
def encode\_property\_type(prop\_type):  
 mapping = {  
 'Konut': 0,  
 'İş Yeri': 1,  
 'Arsa': 2,  
 'Daire': 3  
 }  
 return mapping.get(prop\_type, 0)  
  
def decode\_property\_type(encoded\_type):  
 mapping = {  
 0: 'Konut',  
 1: 'İş Yeri',  
 2: 'Arsa',  
 3: 'Daire'  
 }  
 return mapping.get(encoded\_type, 'Daire')  
  
  
def encode\_city(city):  
 mapping = {  
 'İstanbul': 0, 'Ankara': 1, 'İzmir': 2,  
 'Muğla': 3, 'Antalya': 4, 'Bursa': 5, 'Mersin': 6  
 }  
 return mapping.get(city, 0)  
  
  
def encode\_heating\_type(heating):  
 mapping = {  
 'Fancoil': 0, 'Merkezi': 1, 'Doğalgaz': 2,  
 'Klima': 3, 'Soba': 4, 'Yok': 5  
 }  
 return mapping.get(heating, 0)  
  
  
def encode\_district(district\_name):  
 return hash(district\_name) % 100  
  
  
def encode\_neighborhood(neighborhood\_name):  
 return hash(neighborhood\_name) % 50  
  
  
def calculate\_demo\_price(size, room\_count, building\_age, city, district):  
 base\_price = 1000000  
 predicted\_price = (  
 base\_price +  
 (size \* 45000) +  
 (room\_count \* 250000) -  
 (building\_age \* 18000) +  
 (hash(district) % 200000 - 100000)  
 )  
  
 city\_multipliers = {  
 'İstanbul': 2.5, 'Ankara': 1.8, 'İzmir': 1.7,  
 'Antalya': 1.6, 'Bursa': 1.4, 'Muğla': 1.3, 'Mersin': 1.2  
 }  
 city\_multiplier = city\_multipliers.get(city, 1.0)  
 predicted\_price \*= city\_multiplier  
 predicted\_price \*= np.random.uniform(0.9, 1.1)  
  
 return max(predicted\_price, 500000)  
  
  
with st.sidebar:  
 st.markdown("# 🏢 О приложении")  
 st.markdown("""  
 ### Предсказание цен для недвижимости  
  
 \*\*Функции:\*\*  
 - 🎯 Предсказание типа недвижимости  
 - 💰 Оценка стоимости объекта  
 - 📋 История расчетов  
 - 💱 Конвертация валют  
  
 \*\*Используемые модели:\*\*  
 - Stacking Ensemble для цены  
 - Random Forest для классификации  
 """)  
  
 st.markdown("### 💱 Курсы валют")  
 for currency, rate in EXCHANGE\_RATES.items():  
 if currency != 'RUB':  
 st.write(f"1 {currency} = {rate} ₽")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()