```python
import pandas as pd
import numpy as np

class RideAnalyzer:
    def __init__(self, data_path):
        self.df = self.load_and_preprocess(data_path)
        self.model = self.train_model()
        self.analysis_results = self.analyze_rides()

    def load_and_preprocess(self, data_path):
        df = pd.read_csv(data_path, sep=';', encoding='utf-8')
        df['start_coords'] = df['Coordenadas Inicio'].apply(lambda x: tuple(map(float, str(x).split(','))))
        df['end_coords'] = df['Coordenadas Fim'].apply(lambda x: tuple(map(float, str(x).split(','))))


        for index, row in df.iterrows():
            if pd.isna(row['Distancia (km)']) or pd.isna(row['Tempo (min)']):
                try:
                    distance = geodesic(row['start_coords'], row['end_coords']).km
                    time = distance * 2
                    df.at[index, 'Distancia (km)'] = distance
                    df.at[index, 'Tempo (min)'] = time
                except:
                    df.at[index, 'Distancia (km)'] = np.nan
                    df.at[index, 'Tempo (min)'] = np.nan


        df = df.dropna(subset=['Distancia (km)', 'Tempo (min)'])
        return df

    def train_model(self):
        y = self.df['Distancia (km)'] * 1.5 + self.df['Tempo (min)'] * 0.5 + 5
        X = self.df[['Distancia (km)', 'Tempo (min)']]
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        model = RandomForestRegressor(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)
        return model

    def analyze_rides(self):
        """Analisa as corridas para encontrar extremos"""
        if len(self.df) == 0:
            return None

        min_time_idx = self.df['Tempo (min)'].idxmin()
        max_time_idx = self.df['Tempo (min)'].idxmax()
        min_dist_idx = self.df['Distancia (km)'].idxmin()
        max_dist_idx = self.df['Distancia (km)'].idxmax()

        return {
            'shortest_time': self.df.loc[min_time_idx].to_dict(),
            'longest_time': self.df.loc[max_time_idx].to_dict(),
            'shortest_distance': self.df.loc[min_dist_idx].to_dict(),
            'longest_distance': self.df.loc[max_dist_idx].to_dict(),
            'stats': {
                'avg_time': self.df['Tempo (min)'].mean(),
                'avg_distance': self.df['Distancia (km)'].mean(),
                'total_rides': len(self.df)
            }
        }

    def get_extreme_rides(self):
        """Retorna as corridas com tempos e distâncias extremos"""
        return {
            'Corrida mais rápida': {
                'Tempo (min)': self.analysis_results['shortest_time']['Tempo (min)'],
                'Distancia (km)': self.analysis_results['shortest_time']['Distancia (km)'],
                'Origem': self.analysis_results['shortest_time']['Coordenadas Inicio'],
                'Destino': self.analysis_results['shortest_time']['Coordenadas Fim']
            },
            'Corrida mais demorada': {
                'Tempo (min)': self.analysis_results['longest_time']['Tempo (min)'],
                'Distancia (km)': self.analysis_results['longest_time']['Distancia (km)'],
                'Origem': self.analysis_results['longest_time']['Coordenadas Inicio'],
                'Destino': self.analysis_results['longest_time']['Coordenadas Fim']
            },
```

```
            'Corrida mais curta': {
                'Tempo (min)': self.analysis_results['shortest_distance']['Tempo (min)'],
                'Distancia (km)': self.analysis_results['shortest_distance']['Distancia (km)'],
                'Origem': self.analysis_results['shortest_distance']['Coordenadas Inicio'],
                'Destino': self.analysis_results['shortest_distance']['Coordenadas Fim']
            },
            'Corrida mais longa': {
                'Tempo (min)': self.analysis_results['longest_distance']['Tempo (min)'],
                'Distancia (km)': self.analysis_results['longest_distance']['Distancia (km)'],
                'Origem': self.analysis_results['longest_distance']['Coordenadas Inicio'],
                'Destino': self.analysis_results['longest_distance']['Coordenadas Fim']
            },
            'Estatísticas': self.analysis_results['stats']
        }

if __name__ == "__main__":
    try:
        analyzer = RideAnalyzer("calculo_rotas.csv")
        extremes = analyzer.get_extreme_rides()

        print("=== Análise de Corridas Extremas ===")
        print(f"\n1. Corrida mais RÁPIDA (menor tempo):")
        print(f"   - Tempo: {extremes['Corrida mais rápida']['Tempo (min)']:.2f} min")
        print(f"   - Distância: {extremes['Corrida mais rápida']['Distancia (km)']:.2f} km")
        print(f"   - Origem: {extremes['Corrida mais rápida']['Origem']}")
        print(f"   - Destino: {extremes['Corrida mais rápida']['Destino']}")

        print(f"\n2. Corrida mais DEMORADA (maior tempo):")
        print(f"   - Tempo: {extremes['Corrida mais demorada']['Tempo (min)']:.2f} min")
        print(f"   - Distância: {extremes['Corrida mais demorada']['Distancia (km)']:.2f} km")
        print(f"   - Origem: {extremes['Corrida mais demorada']['Origem']}")
        print(f"   - Destino: {extremes['Corrida mais demorada']['Destino']}")

        print(f"\n3. Corrida mais CURTA (menor distância):")
        print(f"   - Distância: {extremes['Corrida mais curta']['Distancia (km)']:.2f} km")
        print(f"   - Tempo: {extremes['Corrida mais curta']['Tempo (min)']:.2f} min")
        print(f"   - Origem: {extremes['Corrida mais curta']['Origem']}")
        print(f"   - Destino: {extremes['Corrida mais curta']['Destino']}")

        print(f"\n4. Corrida mais LONGA (maior distância):")
        print(f"   - Distância: {extremes['Corrida mais longa']['Distancia (km)']:.2f} km")
        print(f"   - Tempo: {extremes['Corrida mais longa']['Tempo (min)']:.2f} min")
        print(f"   - Origem: {extremes['Corrida mais longa']['Origem']}")
        print(f"   - Destino: {extremes['Corrida mais longa']['Destino']}")

        print("\n=== Estatísticas Gerais ===")
        print(f"Total de corridas analisadas: {extremes['Estatísticas']['total_rides']}")
        print(f"Tempo médio das corridas: {extremes['Estatísticas']['avg_time']:.2f} min")
        print(f"Distância média das corridas: {extremes['Estatísticas']['avg_distance']:.2f} km")

    except Exception as e:
        print(f"Erro ao executar a análise: {str(e)}")
```

```
/usr/local/lib/python3.11/dist-packages/geopy/point.py:472: UserWarning: Latitude normalization has been prohibited in the newer version
  return cls(*args)
=== Análise de Corridas Extremas ===

1. Corrida mais RÁPIDA (menor tempo):
   - Tempo: 0.00 min
   - Distância: 0.00 km
   - Origem: -25.5373683, -49.0911459
   - Destino: -25.5373683, -49.0911459

2. Corrida mais DEMORADA (maior tempo):
   - Tempo: 13173.81 min
   - Distância: 6586.91 km
   - Origem: -82.679786, -35.9533409
   - Destino: -23.6103916, -46.4802573

3. Corrida mais CURTA (menor distância):
   - Distância: 0.00 km
   - Tempo: 0.00 min
   - Origem: -25.5373683, -49.0911459
   - Destino: -25.5373683, -49.0911459

4. Corrida mais LONGA (maior distância):
   - Distância: 6586.91 km
   - Tempo: 13173.81 min
   - Origem: -82.679786, -35.9533409
```

```
    - Destino: -23.6103916, -46.4802573

=== Estatísticas Gerais ===
Total de corridas analisadas: 16822
Tempo médio das corridas: 44.49 min
Distância média das corridas: 23.15 km
```