```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics.pairwise import cosine_similarity
import datetime,csv,os


astrologers = pd.DataFrame([
    {"name": "Rahul Sharma", "expertise": "Vedic Astrology", "experience": 12, "rating": 4.9, "price": 500,
     "mode": "Chat, Call", "language": "Hindi, English,Telugu", "specialization": "Love, Career"},
    {"name": "Ms. Meera ", "expertise": "Tarot, Numerology", "experience": 8, "rating": 4.8, "price": 300,
     "mode": "Video", "language": "Telugu,English", "specialization": "Health, Spiritual"},
    {"name": "changanti Iyer", "expertise": "Nunerology, Vedic", "experience": 15, "rating": 4.95, "price": 800,
     "mode": "Call", "language": "Tamil, English", "specialization": "Marriage, Business"},
    {"name": "Vani", "expertise": "Numerology, Tarot", "experience": 5, "rating": 4.7, "price": 250,
     "mode": "Chat, Video", "language": "English, Hindi", "specialization": "Love, Anxiety"},
    {"name": "Acharya Dev", "expertise": "Vedic, Numerology", "experience": 10, "rating": 4.85, "price": 450,
     "mode": "Call, Chat", "language": "Hindi,Telugu", "specialization": "Finance, Career"}
])
astrologers.head()
```

	name	expertise	experience	rating	price	mode	language	specialization
0	Rahul Sharma	Vedic Astrology	12	4.90	500	Chat, Call	Hindi, English,Telugu	Love, Career
1	Ms. Meera	Tarot, Numerology	8	4.80	300	Video	Telugu,English	Health, Spiritual
2	changanti Iyer	Nunerology, Vedic	15	4.95	800	Call	Tamil, English	Marriage, Business
3	Vani	Numerology, Tarot	5	4.70	250	Chat, Video	English, Hindi	Love, Anxiety

Next steps: Generate code with astrologers | View recommended plots | New interactive sheet

```
class HoroscopeBot:
    def __init__(self, df):
        self.df = df

    def prepare_data(self, sign, category):
        return self.df[
            (self.df['sign'].str.lower() == sign.lower()) &
            (self.df['category'].str.lower() == category.lower())
        ].copy()

    def build_model(self, texts):
        vectorizer = TfidfVectorizer(stop_words='english')
        tfidf = vectorizer.fit_transform(texts)
        model = NearestNeighbors(n_neighbors=3, metric='cosine')
        model.fit(tfidf)
        return vectorizer, model

    def predict_all_signs(self, category, prompt, n_predictions=1):
        signs = [
            "Aries", "Taurus", "Gemini", "Cancer", "Leo", "Virgo",
            "Libra", "Scorpio", "Sagittarius", "Capricorn", "Aquarius", "Pisces"
        ]
        results = []
        for sign in signs:
            prediction = self.predict(sign, category, prompt, n_predictions)
            results.append(f"\n {sign} Prediction:\n{prediction}\n")
        return "\n".join(results)

    def predict(self, sign, category, prompt, n_predictions=3):
        data = self.prepare_data(sign, category)
        if data.empty:
            return "Sorry, no predictions available for that sign/category."

        vectorizer, model = self.build_model(data['horoscope'])
        query_vec = vectorizer.transform([prompt])
        _, indices = model.kneighbors(query_vec, n_neighbors=min(n_predictions, len(data)))

        predictions = []
        for i, idx in enumerate(indices[0]):
            match = data.iloc[idx]
            text = f"\n Prediction {i+1} for {sign.title()} ({category.title()}):\n{match['horoscope']}\n Based on: {match['date'].date()
```

```
            predictions.append(text)

        return "\n".join(predictions)
```

init(self, df):

Input: A Pandas DataFrame (df) containing horoscopes.

Stores: This DataFrame into the bot's internal state for future use.

prepare_data(self, sign, category):

Filters the horoscope dataset for a specific zodiac sign and category (e.g., love, career).

Returns: A filtered DataFrame with only matching rows.

Purpose: Helps to narrow down the search space.

build_model(self, texts):

Input: A list/series of horoscope texts (historical predictions).

Process:Transforms them using TF-IDF (TfidfVectorizer) to convert text into numerical vectors. Fits a NearestNeighbors model to perform similarity-based search using cosine distance.

Returns: The TF-IDF vectorizer and the trained similarity model.

predict(self, sign, category, prompt, n_predictions=3):

Goal: Predict relevant horoscopes based on user's sign and mood.

Steps:

1. Prepares data using prepare_data.
2. If no matching data, returns an error message.
3. Builds a model on the filtered horoscope texts.
4. Transforms the user's prompt into a vector.
5. Finds the n_predictions most similar past horoscopes.
6. Returns formatted predictions, including date.

predict_all_signs(self, category, prompt, n_predictions=1):

Purpose: Runs predict(...) for all 12 zodiac signs.Useful when user wants general predictions for everyone (not just their own sign).

Returns: Combined predictions for all signs.

```
def recommend_astrologers(user_interest, preferred_lang, top_n=3):
    tfidf = TfidfVectorizer()
    vecs = tfidf.fit_transform(astrologers['specialization'])
    user_vec = tfidf.transform([user_interest])
    scores = cosine_similarity(user_vec, vecs).flatten()
    astrologers['match_score'] = scores

    filtered = astrologers[astrologers['language'].str.contains(preferred_lang, case=False)]
    ranked = filtered.sort_values(by=['match_score', 'rating'], ascending=False).head(top_n)

    result = "\n Top Astrologer Recommendations:\n"
    for i, row in ranked.iterrows():
        result += f"\n {row['name']} ({row['expertise']}) - Rating: {row['rating']}/5\n"
        result += f"Language: {row['language']} | Mode: {row['mode']} | ₹{row['price']}\n"
        result += f"Specializes in: {row['specialization']}\n"
    return result



def get_sun_sign(day, month, year):
    if (month == 12 and day >= 22) or (month == 1 and day <= 19):
        return "Capricorn"
    elif (month == 1 and day >= 20) or (month == 2 and day <= 18):
        return "Aquarius"
    elif (month == 2 and day >= 19) or (month == 3 and day <= 20):
        return "Pisces"
    elif (month == 3 and day >= 21) or (month == 4 and day <= 19):
        return "Aries"
    elif (month == 4 and day >= 20) or (month == 5 and day <= 20):
        return "Taurus"
    elif (month == 5 and day >= 21) or (month == 6 and day <= 20):
        return "Gemini"
```

```
        elif (month == 6 and day >= 21) or (month == 7 and day <= 22):
            return "Cancer"
        elif (month == 7 and day >= 23) or (month == 8 and day <= 22):
            return "Leo"
        elif (month == 8 and day >= 23) or (month == 9 and day <= 22):
            return "Virgo"
        elif (month == 9 and day >= 23) or (month == 10 and day <= 22):
            return "Libra"
        elif (month == 10 and day >= 23) or (month == 11 and day <= 21):
            return "Scorpio"
        elif (month == 11 and day >= 22) or (month == 12 and day <= 21):
            return "Sagittarius"


    def run_chatbot():
        # Load horoscope dataset
        df = pd.read_csv("/content/sample_data/horoscope_saved.csv", on_bad_lines='skip')
        df['date'] = pd.to_datetime(df['date'], format='%Y%m%d', errors='coerce')
        df = df.dropna(subset=['horoscope', 'sign', 'category', 'date'])
        bot = HoroscopeBot(df)

        print("\n Welcome to the AI Horoscope and Astrologer Bot!")
        # Setup user log storage
        user_log_file = "user_interactions.csv"
        log_fields = ["name", "dob", "time", "place", "zodiac", "category", "prompt", "language"]
        if not os.path.exists(user_log_file):
            with open(user_log_file, mode='w', newline='') as f:
                writer = csv.DictWriter(f, fieldnames=log_fields)
                writer.writeheader()

        while True:
            dob_str = input("Enter your date of birth (YYYY-MM-DD) or type 'exit' to quit: ").strip()
            if dob_str.lower() == 'exit':
                print(" May the stars guide you. Goodbye!")
                break
            time_str = input("Enter your time of birth (HH:MM, 24hr format): ").strip()
            place = input("Enter your place of birth (City name): ").strip()
            if dob_str.lower() == 'exit':
                print("\n May the stars guide you. Goodbye!")
                break

            try:
                dob = datetime.datetime.strptime(f"{dob_str} {time_str}", "%Y-%m-%d %H:%M")
                zodiac_sign = get_sun_sign(dob.day, dob.month, dob.year)
                print(f"Your zodiac sign is: {zodiac_sign}")
            except ValueError:
                print("Invalid date or time format. Please use YYYY-MM-DD for date and HH:MM (24-hour) for time.")
                continue

            category = input("Choose a category [general, career, love, wellness, finance]: ").strip().lower()
            prompt = input("Briefly describe your current mood or situation (optional): ").strip()
            if not prompt:
                prompt = "What does the universe have in store?"

            lang = input("Preferred language (Hindi/English/Tamil/Telugu): ").strip().capitalize()

            all_signs = input("Would you like predictions for all zodiac signs? (yes/no): ").strip().lower()
            if all_signs == 'yes':
                prediction = bot.predict_all_signs(category, prompt, n_predictions=1)
            else:
                prediction = bot.predict(zodiac_sign, category, prompt, n_predictions=3)
            print(prediction)

            # Store user interaction
            user_data = {
                "name": input("Enter your name: ").strip(),
                "dob": dob_str,
                "time": time_str,
                "place": place,
                "zodiac": zodiac_sign,
                "category": category,
                "prompt": prompt,
                "language": lang
            }
            with open(user_log_file, mode='a', newline='') as f:
                writer = csv.DictWriter(f, fieldnames=log_fields)
                writer.writerow(user_data)
```

```
        print(recommend_astrologers(category, lang, top_n=3))

# ------------------ Run the App ------------------ #
if __name__ == "__main__":
    run_chatbot()
```

```
 Welcome to the AI Horoscope and Astrologer Bot!
Enter your date of birth (YYYY-MM-DD) or type 'exit' to quit: 1997-12-12
Enter your time of birth (HH:MM, 24hr format): 23:09
Enter your place of birth (City name): kkd
Your zodiac sign is: Sagittarius
Choose a category [general, career, love, wellness, finance]: wellness
Briefly describe your current mood or situation (optional): fine
Preferred language (Hindi/English/Tamil/Telugu): Telugu
Would you like predictions for all zodiac signs? (yes/no): no

 Prediction 1 for Sagittarius (Wellness):
The wonderful quality of empathy that you represent is only as good as it feels to you. That is, if you are not able to give someone els
🕑 Based on: 2020-08-02

 Prediction 2 for Sagittarius (Wellness):
Your dreams and intuition meet head-on with your will to succeed and your drive. But maybe you don't really have a goal in mind that wou
🕑 Based on: 2021-02-28

 Prediction 3 for Sagittarius (Wellness):
One of the best ways to feel free of negative energy is to make an agreement with yourself to take a break from reading or listening to
🕑 Based on: 2020-07-27
Enter your name: sri

 Top Astrologer Recommendations:

 Rahul Sharma (Vedic Astrology) - Rating: 4.9/5
Language: Hindi, English,Telugu | Mode: Chat, Call | ₹500
Specializes in: Love, Career

 Acharya Dev (Vedic, Numerology) - Rating: 4.85/5
Language: Hindi,Telugu | Mode: Call, Chat | ₹450
Specializes in: Finance, Career

 Ms. Meera  (Tarot, Numerology) - Rating: 4.8/5
Language: Telugu,English | Mode: Video | ₹300
Specializes in: Health, Spiritual

Enter your date of birth (YYYY-MM-DD) or type 'exit' to quit: exit
 May the stars guide you. Goodbye!
```

Start coding or generate with AI.